

Tame your Work Flow

*How Dr. Goldratt of "The Goal" would apply
the **Theory of Constraints** to rethink
knowledge-work management*



**Steve Tendon
Daniel Doiron**



Forewords by
Eli Schragenheim
Daniel S. Vacanti

Tame your Work Flow

How Dr. Goldratt of “The Goal” would apply the Theory of Constraints to rethink knowledge-work management (First Edition)

Steve Tendon and Daniel Doiron

This book is for sale at <http://leanpub.com/workflow>

This version was published on 2020-02-10

ISBN 978-99957-44-09-0



This is a **TameFlow Press** book.

TameFlow Press - a business unit of **TameFlow Consulting Limited** - brings you the most current information and cutting edge ideas about the *TameFlow Approach* to help you develop your own breakthrough *performance-innovation* for your organization.

For more information about the *TameFlow Approach*, visit: <https://tameflow.com>.

The “TameFlow” mark and the TameFlow logo are ® Registered with the US Patent & Trademark Office.

© 2020 Copyright by TameFlow Press / TameFlow Consulting Limited

Tweet This Book!

Please help Steve Tendon and Daniel Doiron by spreading the word about this book on [Twitter](#)!

The suggested hashtag for this book is [#tameflow](#).

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

[#tameflow](#)

Contents

| | |
|--|-----------|
| About this Book | 1 |
| Free Bonuses! | 1 |
| What is this book about? | 1 |
| Is this book for you? | 2 |
| What you need to know before reading this book | 3 |
| Scope of applicability the TameFlow Approach | 3 |
| Relevance of this book | 5 |
| How this book is organized | 6 |
| Part 1 - Getting More for Nothing with Flow Efficiency | 6 |
| Part 2 - Acting on the Archimedean Lever that Boosts Performance: the Constraint | 6 |
| Part 3 - Making Accounting Make Money | 6 |
| Part 4 - Maximizing Business Value in Knowledge-Work | 7 |
| Part 5 - Preparing for High Performance Execution and Governance | 7 |
| Part 6 - Achieving High Performance Execution and Governance | 8 |
| Part 7 - Igniting High-Performance | 8 |
| Credits | 8 |
| Typographical Conventions | 8 |
| Disclaimers | 9 |
| Author Views | 9 |
| Quotations | 9 |
| Testimonials | 9 |
| Acknowledgements | 9 |
| Dedication | 10 |
| Forewords | 11 |
| Foreword by Eli Schragenheim | 12 |
| Foreword by Daniel S. Vacanti | 13 |
| Introduction | 14 |
| Introduction by Steve | 14 |
| Introduction by Daniel | 16 |
| Prologue | 17 |
| The Story of Herbie | 17 |
| Herbie and Your Work Flow | 20 |
| The Five Focusing Steps | 21 |

CONTENTS

| | |
|--|-----------|
| Step 1: Identify the Constraint - "Herbieeeee!" | 21 |
| Step 2: Exploit the Constraint - "C'mon Herbie! Speed up!" | 21 |
| Step 3: Subordinate to the Constraint - "Everybody stays behind Herbie!" | 22 |
| Step 4: Elevate the Constraint - "Everybody carries a piece of Herbie's gear!" | 22 |
| Step 5: Repeat! | 22 |
| The Unstated Step 0 | 22 |
| The Extra Step 6 | 22 |
| The Secret of all Steps | 22 |
| Daniel Gagnon says... | 24 |
| PART 1—Getting More for Nothing with Flow Efficiency | 25 |
| John Coleman says... | 26 |
| 1—The Power of Explicit Mental Models | 27 |
| What is this <i>TameFlow</i> , anyway? | 27 |
| What is <i>Flow</i> and <i>Throughput</i> ? | 28 |
| Decision-Making through Explicit Mental Models | 29 |
| A Mental Model to Explain the Business Value of <i>Flow</i> | 30 |
| The Desire to Increase Performance | 31 |
| Increasing Performance through Effort Bears Many Downsides | 32 |
| The Rate of <i>Demand</i> is Important Too | 32 |
| The Difference Between Demand and Delivery is Even More Important | 34 |
| Why Companies are Always Overburdened | 35 |
| The Hideous Effects of Multitasking | 36 |
| The Business Value of Matching Delivery to Demand | 37 |
| Thinking about the Demand Line instead of the Delivery Line | 38 |
| The Quest for Stability and Predictability | 39 |
| Unintended Consequences of Good Intentions | 40 |
| Takeaways | 40 |
| Chris Matts says... | 42 |
| Troy Magennis says... | 42 |
| 2—Postpone Commitment and Limit Work in Process | 43 |
| Limiting Work in Process | 44 |
| Postpone Commitment - But We Cannot Wait! | 45 |
| Takeaways | 49 |
| Kurt Häusler says... | 50 |
| 3—Flow Efficiency, Little's Law and Economic Impact | 51 |
| Touch Time and Wait Time | 51 |
| Controlling the Wait Time | 52 |
| Getting the <i>Flow Efficiency</i> Calculation Right | 53 |
| Wait Time and Options | 53 |
| The Perspective of the <i>Work Item</i> | 54 |
| Thinking about <i>Touch Time</i> and <i>Wait Time</i> in Practice | 54 |
| Work Faster or Deliver Sooner? | 56 |
| What is <i>Flow Efficiency</i> ? | 57 |
| <i>Flow Efficiency</i> , Multitasking and Other Time Snatchers | 58 |

CONTENTS

| | |
|---|------------|
| Beware of Sprints - A Big Time Snatcher | 59 |
| What is Little's Law? | 60 |
| Assumptions of Little's Law | 61 |
| Stability and Mental Models | 62 |
| Little's Law and the Theory of Constraints | 62 |
| Little's Law and Flow Efficiency | 63 |
| Economic Impact | 63 |
| Takeaways | 65 |
| Marcus Hammarberg says... | 66 |
| 4—Utility of Flawed Mental Models | 67 |
| Flawed Mental Models are not Necessarily Bad | 67 |
| A Good Wrong Reason to Accept a Flawed Mental Model | 68 |
| Misunderstanding the Effect of Wait Time on Little's Law | 68 |
| Alleged Economic Benefits for the Wrong Reasons | 69 |
| Why the Flawed Model has Utility | 70 |
| The Shortcomings of the Model | 72 |
| Reducing Wait Time does not Increase Throughput (Directly) | 72 |
| The Real Beneficial “Side Effect” of Improving on Wait Time | 72 |
| The Importance of Proper Metrics | 75 |
| Effects and Limits of “Deliver Sooner” | 75 |
| Reducing Wait Time vs Touch Time | 76 |
| Effect of Removing Multitasking | 77 |
| Overall Effect of Reducing Wait Time | 77 |
| When to Switch from Wait Time to Touch Time | 82 |
| Improving Flow Efficiency is a Low Hanging Fruit | 83 |
| Takeaways | 83 |
| Daniel Couture says... | 85 |
| PART 2—Acting on the Archimedean Lever that Boosts Performance: the Constraint | 86 |
| Al Shalloway says... | 87 |
| 5—Where to Focus Improvement Efforts | 88 |
| On Bottlenecks and Constraints | 90 |
| Bottlenecks are not Constraints | 90 |
| Constraints are not only Bottlenecks - There are Other Kinds of Constraints | 92 |
| Where to Improve, Where to Invest | 93 |
| Takeaways | 104 |
| Robert Newbold says... | 106 |
| Corey Ladas says... | 106 |
| PART 3—Making Accounting Make Money | 107 |
| Bob Sproull says... | 108 |
| Russell Eirich says... | 108 |
| 6—Introduction to Throughput Accounting and Culture | 109 |
| Kanban Models and Culture | 109 |
| Accounting Conflicts | 110 |

CONTENTS

| | |
|--|------------|
| Smart Money at the <i>Constraint</i> | 112 |
| Unanimity or Consensus Based decision-making? | 112 |
| <i>Throughput Accounting Basics</i> | 113 |
| Takeaways | 114 |
| Johanna Rothman says... | 115 |
| 7—Accounting F(r)iction | 116 |
| The Story of Daniel at TRUSTNEWS | 116 |
| Cost Control and <i>Flow Accounting</i> at TRUSTNEWS | 116 |
| Khenbish is Back! | 119 |
| Absence of <i>Constraints Management</i> | 119 |
| Expertise Mix between in house and consultants | 123 |
| The <i>Throughput Accounting Report</i> | 125 |
| Takeaways | 126 |
| “Dr. Lisa” Lang says... | 128 |
| Clarke Ching says... | 128 |
| 8—Show Me the Money | 129 |
| The <i>Financial Throughput Report</i> | 131 |
| The “GOAL” Line | 136 |
| <i>Throughput Accounting</i> - Pain Relief for the C-Levels | 138 |
| Takeaways | 139 |
| Savio Neville Spiteri says... | 141 |
| PART 4—Maximizing Business Value in Knowledge-Work | 142 |
| Stefan Willuda says... | 143 |
| 9—Constraints in the Work Flow and in the Work Process | 144 |
| VUCA and PEST | 145 |
| <i>Constraint</i> in the Work Flow and <i>Constraint</i> in the Work Process | 147 |
| Where is the <i>Constraint</i> ? | 147 |
| The “Painting Gadgets” Example | 148 |
| From <i>Work Flow</i> to <i>Work Process</i> | 149 |
| Shape and Form of Demand | 150 |
| Relation to <i>Special Cause Variation</i> and <i>Common Cause Variation</i> | 151 |
| Takeaways | 152 |
| Mario Latreille says... | 153 |
| 10—Understanding PEST Environments | 154 |
| Overview of the TameFlow Simulation | 155 |
| Simulating a PEST Environment: Initial Setup | 156 |
| The <i>Work Flow</i> and <i>Work Process</i> for a Single Project and Four Teams | 158 |
| Round 1 - The Warm Up | 159 |
| Round 2 - First Estimation | 160 |
| Round 3 - Two Projects and Conflicts of Interests Immediately Emerge | 160 |
| Skipping the Line Becomes the Norm | 162 |
| Bonuses, Rewards and Budgets Create Conflicts of Interests | 162 |
| Round 4 - Ten Projects and Three <i>Product Owners</i> | 163 |

| | |
|--|------------|
| Takeaways | 165 |
| Etienne Du Plooy says... | 166 |
| 11—Finding the Constraint in PEST Environments | 167 |
| Impact of the Shape and Form of Demand | 167 |
| Expose all (Virtual) Queues of Work Load without WIP Limit Distortions | 168 |
| Focus on the <i>Constraint</i> in the <i>Work Flow</i> First, then on the <i>Constraint</i> in the <i>Work Process</i> | 174 |
| Takeaways | 175 |
| Hermann Hyytiälä says... | 176 |
| Sanjeev Gupta says... | 176 |
| 12—Drum-Buffer-Rope Scheduling | 177 |
| Reflections on <i>Column WIP Limits</i> | 177 |
| Introducing <i>Drum-Buffer-Rope</i> in the <i>Work Flow</i> | 179 |
| <i>Drum-Buffer-Rope</i> in Practice | 180 |
| <i>Drum-Buffer-Rope</i> Portfolio Kanban Boards | 182 |
| Location vs. Visualization of the <i>Constraint</i> in the <i>Work Flow</i> | 183 |
| The Minimum Number of <i>Work Items</i> in the <i>Buffer</i> | 184 |
| The <i>Constraint</i> in the <i>Work Process</i> of the <i>Constraint</i> in the <i>Work Flow</i> | 185 |
| Takeaways | 185 |
| Matthew Croker says... | 186 |
| 13—Portfolio Prioritization and Selection in PEST Environments | 187 |
| Prioritizing for Business Value | 188 |
| Quantifying the <i>Work Load</i> | 190 |
| Economic Prioritization and Selection via <i>Throughput Rate</i> | 192 |
| <i>Cost of Delay</i> and <i>CD3</i> | 195 |
| Further Notes on <i>Cost of Delay</i> and <i>CD3</i> | 200 |
| The Need for Estimation Practices | 201 |
| Estimation/Forecast of duration, not of due dates | 202 |
| T-Shirt Sizing | 202 |
| Probabilistic Forecasting with <i>Flow Metrics</i> | 202 |
| Advanced Application: Estimate the <i>Flow Time</i> on the <i>Constraint</i> alone | 203 |
| Takeaways | 204 |
| Emiliano Sutil says... | 205 |
| PART 5—Preparing for High Performance Execution and Governance | 206 |
| Wolfram Müller says... | 207 |
| 14—Flow Efficiency, DBR and TameFlow Kanban Boards | 208 |
| New Types of <i>Kanban Boards</i> | 209 |
| Flow Efficiency Board | 211 |
| Full-Kitting | 215 |
| Explicit <i>Waiting for</i> and <i>In Process</i> Columns | 216 |
| Managing <i>Flowbacks</i> | 218 |
| <i>Flow Efficiency Inflation</i> with <i>Flowbacks</i> and Conventional <i>Kanban Boards</i> | 218 |
| <i>Flowbacks</i> with <i>Flow Efficiency Boards</i> | 219 |
| How to Manage <i>Flowbacks</i> in TameFlow | 219 |

CONTENTS

| | |
|--|------------|
| <i>Flow Efficiency Boards and Complex Work Flows</i> | 220 |
| <i>Drum-Buffer-Rope Board</i> | 221 |
| Using <i>Flow Efficiency Boards</i> to Identify the <i>Constraint</i> in the <i>Work Process</i> | 222 |
| From <i>Flow Efficiency Boards</i> to <i>DBR Boards</i> | 223 |
| <i>Buffer Zones</i> and <i>Buffer Signals</i> | 225 |
| <i>The TameFlow Throughput Management Board</i> | 229 |
| Revisiting the Portfolio Board | 229 |
| Acclamation of Idleness | 232 |
| The Unresolvable Conflict between Idleness and Cost Accounting | 232 |
| Cultural Impact of the <i>TameFlow Boards</i> | 233 |
| Psychology of Wait States | 233 |
| Takeaways | 233 |
| Curtis Hibbs says... | 235 |
| 15—Outcomes, Values and Efforts in PEST Environments | 236 |
| The Virtue of Minimalism: the <i>Minimal Outcome-Value Effort</i> (MOVE) | 236 |
| What is a MOVE? | 236 |
| A Unit of Business Outcome | 238 |
| A Unit of <i>Throughput Value</i> | 238 |
| A Unit of Costing and Reporting | 239 |
| A Unit of Work and Delivery | 239 |
| A Small Target-Scope Work Package | 240 |
| The Balance of Two Opposing Forces | 241 |
| Unit of Commitment | 242 |
| Mechanism to Limit <i>Work in Process</i> | 242 |
| Risk Managed via Time and not Scope Adjustments | 242 |
| A Note on Agile | 242 |
| Managing Scope Variation | 243 |
| Target-Scope is not Fixed-Scope | 243 |
| A MOVE View of the Simulation | 244 |
| Takeaways | 245 |
| Martin Nantel says... | 246 |
| 16—Introduction to Execution Management Signals | 247 |
| The Logic of Critical-Chain Project Management (CCPM) | 248 |
| Point Estimates vs. Probability Distributions | 249 |
| Critical Chain Planning | 250 |
| The Critical Chain Plan | 252 |
| The Critical Chain Buffer | 253 |
| <i>Buffer Zones</i> | 255 |
| <i>Buffer Consumption</i> and <i>Buffer Burn Rate</i> | 256 |
| Execution Management Signals | 257 |
| Visual Execution Management and MOVES | 258 |
| The MOVE Buffer | 258 |
| Visual Execution Management | 260 |
| From One Team and One MOVE to One Team and Many MOVES | 262 |
| From One Team and Many MOVES to Many Teams with Many MOVES | 264 |
| Takeaways | 265 |

| | |
|---|------------|
| Michael Küsters says... | 266 |
| 17—Introduction to Full-Kitting | 267 |
| Introduction to Full-Kitting | 267 |
| We don't have time for this! | 267 |
| Capitalizing on Excess Capacity | 268 |
| Seizing a Zero-Cost Benefit | 270 |
| Improving the Work Process | 271 |
| We Cannot Do Big Up-Front Design in a VUCA World and be Agile | 271 |
| Dedicated Roles | 272 |
| The Nature of Full-Kitting | 273 |
| Simulating Full-Kitting on the Client-Side | 273 |
| Simulating Missing Information | 275 |
| Round 1 - Heads or Tails as Bearer of Information | 276 |
| Round 2 - Avoiding Flowbacks | 278 |
| Round 3 - Simulating Full-Kitting on the Team Side | 279 |
| Takeaways | 280 |
| Jerzy Stawicki says... | 281 |
| PART 6—Achieving High Performance Execution and Governance | 282 |
| Vasco Duarte Says... | 283 |
| 18—Full-Kitting as Ongoing Executive Activity | 284 |
| Failure Demand | 284 |
| The Full-Kitting Work Flow | 285 |
| The Backlog Column | 285 |
| The Full-Kit Column | 286 |
| The Prioritization Column | 286 |
| The Ranking Column | 287 |
| The Committed Column | 288 |
| The In Flow Column | 288 |
| Operational Full-Kitting | 289 |
| Takeaways | 289 |
| Daniel Hernández says... | 290 |
| 19—Execution Management in PEST Environments | 291 |
| Many Moving MOVEs | 292 |
| Virtual Integration Points | 292 |
| Planning with Respect to the Constraint | 293 |
| Monitoring the MOVE Buffers' Consumptions | 294 |
| The Constraint Caused by Execution Issues | 296 |
| Full-Kitting with an Eye on the State of Execution | 297 |
| Takeaways | 297 |
| Christophe Achouiantz says... | 299 |
| 20—Operational Governance in PEST Environments | 300 |
| Constraints Everywhere! | 300 |
| Early Detection Signals | 302 |

CONTENTS

| | |
|--|------------|
| Focused Governance | 305 |
| The Meaning of Red | 306 |
| Overload Detection | 307 |
| Thinking is Still Required | 309 |
| Full-Kitting Revisited | 309 |
| Management by Exception - Limit Meetings | 310 |
| Management by Exception - Ad-hoc Meetings | 310 |
| Effective and Focused Standup Meetings with the <i>TameFlow Approach</i> | 311 |
| MOVE Reviews and Retrospectives | 313 |
| Informational Flow | 313 |
| Notes on Cadences | 313 |
| Specific Kanban Cadences | 315 |
| A Vocabulary for <i>Thinking about Constraints</i> | 316 |
| Constraint in the Work Flow | 317 |
| Constraint in the Work Process | 317 |
| Constraint in the Work Execution | 317 |
| Takeaways | 318 |
| Tom Cagley says... | 319 |
| PART 7—Igniting High Performance | 320 |
| Joseph Hurtado Says... | 321 |
| 21—Patterns to Get Started! | 322 |
| A Patterns Based Approach | 322 |
| Baby Steps toward Improvement | 323 |
| Pattern Network Traversal is not Stage-Gated Project Management | 325 |
| Actionable Use of Metrics | 326 |
| Iterative Process of Ongoing Improvement | 326 |
| Pattern 1 - Leave No One Behind | 326 |
| Assignment | 327 |
| Expected Effects | 327 |
| Pattern 2 - One Slice at a Time | 327 |
| Assignment | 328 |
| Expected Effect | 328 |
| Pattern 3 - Set The Goal | 329 |
| The Goal | 329 |
| Critical Success factors | 330 |
| Necessary Conditions | 330 |
| Assignment | 332 |
| Expected Effect | 332 |
| Pattern 4 - Prime the Mindset | 332 |
| Assignment | 334 |
| Expected Effect | 334 |
| Pattern 5 - Measure The Ends and Have a Plot | 334 |
| Assignment | 334 |
| Expected Effect | 334 |
| Pattern 6 - Show the Work | 335 |
| Assignment | 335 |

CONTENTS

| | |
|--|-----|
| Expected Effects | 335 |
| Pattern 7 - Show the Flow | 335 |
| Assignment | 335 |
| Expected Effects | 336 |
| Pattern 8 - Show the Numbers | 336 |
| Assignment | 336 |
| Expected Effects | 336 |
| Pattern 9 - Limit Work in Process | 336 |
| Assignment | 337 |
| Expected Effects | 337 |
| Pattern 10 - Make MOVEs | 337 |
| Assignment | 338 |
| Expected Effects | 338 |
| Pattern 11 - Forecast (or Estimate) | 338 |
| Assignment | 339 |
| Expected Effects | 339 |
| Pattern 12 - Place and Use the <i>Buffers</i> | 339 |
| Assignment | 339 |
| Expected Effects | 339 |
| Pattern 13 - Bake the Whole Cake | 339 |
| Assignment | 340 |
| Expected Effects | 340 |
| Pattern 14 - Set Priorities and Sequences and Get to <i>Full-Kitting</i> | 340 |
| Assignment | 340 |
| Expected Effects | 341 |
| Pattern 15 - Keep Feeding Herbie | 341 |
| Assignment | 341 |
| Expected Effects | 341 |
| Pattern 16 - Catch the Signals | 341 |
| Assignment | 342 |
| Expected Effects | 342 |
| Pattern 17 - Stand Up for a Cause | 342 |
| Assignment | 344 |
| Expected Effects | 344 |
| Pattern 18 - Swarm the Block | 344 |
| Assignment | 345 |
| Expected Effects | 345 |
| Pattern 19 - Bubble Baths | 345 |
| Assignment | 345 |
| Expected Effect | 345 |
| Pattern 20 - Keep Reason Logs | 346 |
| Assignment | 346 |
| Expected Effects | 346 |
| Pattern 21 - Look Back at the Roots | 346 |
| Assignment | 347 |
| Expected Effects | 347 |
| Pattern 22 - Become a Focused Company | 347 |
| Assignment | 348 |

CONTENTS

| | |
|--|------------|
| Expected Effects | 348 |
| Takeaways | 348 |
| Stuart Burchill says... | 349 |
| Epilogue - It is Never “Done!” | 350 |
| What Have we “Done?” | 350 |
| The Prevalence of <i>Mental Models</i> | 352 |
| Mindsets and Attitudes to Win in a VUCA World | 352 |
| Matt Barcomb says... | 356 |
| Minton Brooks says... | 356 |
| Bibliography | 357 |
| About the Authors | 358 |
| Steve Tendon | 359 |
| Daniel Doiron | 360 |
| Resources | 361 |
| Community | 361 |
| Professional Service, Executive Education and Training | 361 |
| Hyper-Productive Knowledge-Work Performance | 362 |
| The Essence of TameFlow | 363 |
| TameFlow Chronicles 2011-2015 | 364 |
| TameFlow Patterns | 365 |
| Help with TameFlow | 366 |
| Free Bonuses! | 367 |

About this Book

Free Bonuses!

Thank you for looking into this book. The book has associated a number of free bonus resources that you may access by visiting:

<https://tameflow.com/tame-your-work-flow-free-bonus/>

You will find:

- Free copy of the book *The TameFlow Games*
- Free copy of the book *TameFlow Chronicles 2011-2015*
- Free copy of the book *TameFlow Patterns*

What is this book about?

This book is a first in the history of modern management methods. It took inspiration from the time when the *Kanban Method* and *Theory of Constraints* coexisted. The *Kanban Method* then decided to move in a different direction and not only distanced itself from, but also misapplied the original teachings of Dr. Eliyahu Goldratt, the founder of the *Theory of Constraints*.

This book is a premiere in applying the clarity of thinking of Dr. Eliyahu Goldratt to the domain of knowledge-work management. Hence the book's subtitle: "*How Dr. Goldratt of 'The Goal' would apply the Theory of Constraints to rethink knowledge-work management.*"

Kanban practitioners live in an idealized and simplified view where the belief is that a "constraint" will make itself visible by queues and starvation columns on a *Kanban Board*. Instead, this book will demonstrate that identifying the real *Constraint* in a complex organization is not that straightforward and simple. The book will show in detail how to contend with what we call *PEST* environments, where we have to deal with multiple **Projects** or **Products**, **Events** (deadlines), **Stakeholders** and **Teams**.

We will discover that the *Constraint* can exist in different places: in the *Work Flow*, in the *Work Process* and in the *Work Execution*; and that we need appropriate techniques to identify and distinguish between the particular cases.

Furthermore, the book will not only show how to apply the ideas of *Constraints Management* but also how to relate them to actual business results by means of *Throughput Accounting*. We will learn to use the techniques of *Throughput Accounting* to make better management decisions between different product/service development options, or product/service mixes. We will also see how *Throughput Accounting* can serve as an effective tool to streamline prioritization and sequencing of project portfolios at scale.

We will discover the use of *Buffer Management* - a concept originally found in *Critical Chain Project Management* - to drive management execution and governance of PEST environments. We will see how animated *Bubble Fever Charts* will allow us to remain in control even in the most demanding multi-project situations.

We will explore the overarching mindset of *Management by Exception*, where the prevailing idea is *not* to intervene or interfere with the system, unless there is a reason to do so. Let people work! Regular

meetings will be reduced to the bare minimum. They will never happen at a predetermined cadence, but only when there are clear *Work Execution Signals* that show that there is a need for managerial attention.

We will also discuss in depth the notion that in a constrained system, the non-constrained resources will necessarily have *Excess Capacity* - and how this is *not* a reason to start cost cutting exercises. Instead, we will learn that such *Excess Capacity* can be put to better use to elevate the capacity of the system and produce even better business results.

By accepting that *Excess Capacity* is necessary, we can also appreciate that most of the people in the organization can afford the time to be *idle*. In high performing systems, we will have *people waiting for work*, rather than *work waiting for people*.

We will acknowledge that such idleness provides the opportunity to cultivate more humane working environments; and that it creates the effective space where human interactions can be rediscovered.

We will also appreciate how the use of *Constraints Management* and *Throughput Accounting* enables *unanimous decision making*. Gone are the days of endless meetings negotiating for consensus that never fully happens. By focusing on the *Constraint* and on *Throughput* all different perspectives will be aligned, and will create the actual conditions for the prospering of the *Unity of Purpose* and the *Community of Trust* which are the founding patterns of the *TameFlow Approach* and that are effectively the bedrock of high-performing organizations.

Is this book for you?

Since the *TameFlow Approach* and the *Theory of Constraints* are *Systems Thinking* approaches, they can bear significance for many different roles in our complex and modern organizations.

Thus, this book will appeal to different categories of readers. Here are some perspectives that you will find represented in the book and which can resonate with your viewpoint:

- **Business Management:** You are a business owner, managing director or chief executive and care about increasing the *bottom line* of your business, and you are in charge of "*how we make money here*." And you have to *run* the business today. And *improve* the business. And build *new business* for tomorrow.
- **Middle Management:** You are a business unit director, department head, line manager or team leader and need to coordinate and synchronize your unit's efforts with other units; and in particular you repeatedly find yourself in a "fighting for resources" situation against your peers in other units.
- **Portfolio/Project/Product Management:** You are a PMO manager, programme/portfolio manager, project manager, product manager, product owner who typically needs to "*get more stuff done faster with less resources*."
- **Agile/Scrum/Process/Method/Methodology Management:** You are a process/methodology engineer, a Scrum Master, an Agile Coach, a Consultant in charge of "*how we get stuff done here*."
- **Knowledge-Work:** You are a product developer, designer, architect, engineer, software developer, etc. who "*actually gets the stuff done here*."

This book will allow you to discover new areas of competencies related to: explicit risk management, capacity, detection of special and common cause variation, management by exception and new scheduling, prioritization and sequencing techniques.

Whether you know about the *Theory of Constraints* and *Throughput Accounting* - or not - and would like to see how to apply them to the *Kanban Method*, this book will give you the right guidance. The concepts addressed throughout this book will help you develop business agility, which is understood as "*the ability to change direction at high speed*." And if you are into Agile, you will discover how the ideas of this book

will make you even more agile, because its data-driven focus squarely lies on better execution of work all the while fostering humane, collaborative and creative working conditions.

If you are well versed in the *Kanban Method*, you might be surprised to discover many of its shortcomings and how they can be addressed through the techniques presented in this book.

What you need to know before reading this book

This book is definitely data-driven. The math is not in itself challenging but a good memory of evolving contexts supported by numbers will definitely be helpful.

While you do not require previous knowledge in order to read this book as everything is explained from the outset, if you are familiar with the *Kanban Method*, *Agile*, *Lean*, *Scrum* or the *Theory of Constraints* you will find the material more accessible.

In particular, if you have experience with Agile, keep in mind that our understanding of Agility is “*the ability to change direction at high speed.*” The spirit of Agile is pursued, albeit with means that are different from what typical Agile approaches would employ.

While traditional *Cost Accounting* is mandatory within the scope of doing business, it is not required knowledge for reading the book; but if you have a basic understanding of conventional accounting, you will have a better grasp of how *Financial Flow* is achieved, and how the alternative proposed approach, *Throughput Accounting*, supports daily performance management and decision making.

If you have experience in Project Portfolio Management you will recognize many of the problems that are addressed in the book, though you will be (pleasantly) surprised by the solutions that are proposed.

This book contains many definitions and references when new topics are first mentioned. The definitions are there to establish common - or presumed - understanding of terminology. The references allow you to delve into the details so that you can forge ahead or visit the referenced topics right away.

Keep in mind that this book is presenting many interrelated concepts. You will most likely not read it as a novel, from start to end, except maybe for the first reading. Instead you will jump back and forth from one topic to another, as you discover how all the concepts work together to provide you with a simple, though non-intuitive approach, to manage the challenges of modern knowledge-work.

Scope of applicability the TameFlow Approach

An inevitable question when considering this book is: “*What is it good for?*” And since this book is about the *TameFlow Approach*, corollary questions are “*What is TameFlow? And what is it good for?*”

These questions are fundamental and the answers are both easy and difficult. Easy in the sense that once you have learned about *TameFlow*, it will all appear as *obvious*; but before you go through the learning curve it will not be *self-evident*. It is a bit like the *Egg of Columbus* - the answer is accessible and easy for anyone, once they know it. It is *obvious* in hindsight, but not *self-evident* at the outset.

But why is it difficult to comprehend what the *TameFlow Approach* is about? Because it cannot easily be put in relation to existing prevailing ideas about how to manage knowledge-work. It is different.

In the *Essence of TameFlow* book we read that the *TameFlow Approach*...

- ...does not have *reference processes* or *practices* (like Lean or XP).
- ...does not have *reference roles, artifacts* or *ceremonies* (like Scrum).
- ...does not have *principles* and *values* (like Agile and the Kanban Method).
- ...does not have *maturity levels* (like CMMI and KMM).

- ...does not have a *reference framework* that must be adopted and adapted to any given situation (like Scrum, Nexus, SAFe, LeSS, Scrum@Scale).

While the *TameFlow Approach* might contemplate any of the above elements, it is not in itself defined in terms of such elements.

Such elements might be considered as a matter of inheritance, because the *TameFlow Approach* is easily superimposed on any existing approach defined in such terms, and then those elements would exist in *that* context. The *TameFlow Approach*, in this respect, is similar to the *Kanban Method* which starts from where you are, and then evolves from there. If the starting conditions include any of the above elements, then the *TameFlow Approach* will start with them too. This is a positive trait, because it means you can start thinking in terms of the *TameFlow Approach* regardless of your starting conditions.

So what is this *TameFlow Approach*?

The *TameFlow Approach* is primarily about mindsets, attitudes and *Mental Models* that support *decision making* at all levels in the organization. The “magic” of the *TameFlow Approach* is that it renders all decisions coherent, consistent and aligned toward one direction; thus eliminating conflicts, and instead focusing effort and energy toward what really matters.

The TameFlow Approach is primarily a way of thinking!

Note: A short definition of the *TameFlow Approach* is this: “*TameFlow is a praxis driven holistic and systemic approach to organizational performance innovation based on reference models of thinking and decision making.*” (Source: The Essence of TameFlow, 2016)

Because the *TameFlow Approach* is a *way of thinking*, its scope of applicability is broad.

For example, in light of the popular *Wardley Maps* we might consider a company with its products and services going through different stages of evolution like: genesis, custom, product/rental services, commodity/utility services. The interesting observation is that as this progression evolves, different management approaches might be appropriate. In the genesis and custom stages, where there are a lot of unknowns and discoveries, Agile approaches are fitting. In the more mature phases, Lean is adequate. In the commodity/utility stage, more deterministic methods like Six Sigma fit the bill. In other words there is no one single size that fits all.

Another relevant popular conceptual scheme, the *Cynefin Framework* with its decision-making domains (obvious, complicated, complex, chaotic and disorder), also highlights the need for different management approaches, depending on the relevant domain.

However, the *TameFlow way of thinking* is applicable and valid across the whole spectrum of such domains.

How is this possible?

The answer is *obvious* in hindsight, but not *self-evident* at the outset.

We explain this with the metaphor of *The Jeep, the Jungle and the Journey*.

The *Jeep* is our organization, our team. The way we work can be established in a very well-known, deterministic manner. We want our *Jeep* to be in the best of conditions and responsive to our steering and driving.

The *Jungle* is the business domain we are confronting. It could be chaotic in a genesis stage; or it could be complicated in a commodity/utility stage. It is the terrain that we have to traverse with our *Jeep*.

The *Journey* is how we actually handle our *Jeep* while moving across the *Jungle*, with all the events that might happen during that particular venture.

The *Jungle* can be full of hazards, which we would like to avoid; the *Journey* can be full of surprises that we would like to anticipate if possible, or to be able to sense and respond to with our driving abilities of the *Jeep*.

The *Mental Models* of *TameFlow* can relate to any of those three: the *Jeep*, the *Jungle* or the *Journey*, which in the book correspond to the terms of the *Work Process*, the *Work Flow* and the *Work Execution*.

Note: The *Mental Models* of *TameFlow* are meant to be shared across the entire organization. “*The aim of TameFlow is to allow anybody, at any level of an organization, to exercise judgement about how to decide and act in the current situation.*” (Source: The Essence of *TameFlow*, 2016)

Relevance of this book

The scope of applicability of the *TameFlow Approach* is broad. The *TameFlow Approach* gains this flexibility because it focuses on four flows, namely: *Operational Flow*, *Financial Flow*, *Informational Flow* and *Psychological Flow*.

In this book we will see different aspects of these four flows. Here are a few examples for whom the content of the book will prove very useful:

- You are a top level executive and you are drowning in information overload - with the *Informational Flow* you will receive only relevant information, when it is needed, with the right people involved, and you will be able to make timely quality management decisions.
- You are in charge of Agile and cannot figure out how to get top executives really engaged and supportive of your efforts - with *Financial Throughput* and in particular the techniques of *Throughput Accounting* you can show how operational decisions directly affect the bottom line.
- You are responsible for a number of projects and teams and need ways to coordinate efforts across all projects and teams - with *Operational Flow*, *DBR Scheduling*, and *Management Signals* you will have new tools.
- You are a team leader and need to get your people more engaged and aligned - with the *Psychological Flow* elements you can get your teams into a state of flow.
- You are an Agile Coach and need to “implement” some “scaling” of Agile methods - with the combination of the four flows you will discover that the *TameFlow Approach* can more easily resolve scaling problems (by actually being “scale-less” in virtue of its *Mental Models*).

More important than single instances or examples of relevance is the overarching idea that by studying the *TameFlow Approach* we (and our teams, peers and superiors) will develop new ways of thinking about what we are doing, and that we will start making decisions differently in the light of the *Mental Models* that you will learn.

It is this idea that is the “essence” of the *TameFlow Approach*: sharing the *Mental Models* supporting superior and faster decision making across the entire organization, so that any part of the organization will be enabled and empowered to *make the right decisions at the right time in their particular context*.

How this book is organized

The book is divided into seven parts. Here's a short summary of the topics covered.

Part 1 - Getting More for Nothing with Flow Efficiency

How can your brain deal with this riddle everyday? "*The demand for our services is overwhelming and we must keep it outside the system and work at the best of our capabilities in order to be more productive and deliver faster.*"

The first four chapters of this book will show you how to overcome obstacles to *Flow Efficiency* and challenge one of the most erroneous *Mental Model* - "the sooner we start, the sooner we finish" - we need to overcome on our way to achieve better *Flow Efficiency* for immediate impact on the bottom line.

This is where trust is being "built in." A reliable and stable system has a better chance of delivering what is promised and of being around next year.

Flow Efficiency improvements come at no financial costs and are simply related to decision making as to whether you want to work harder or deliver sooner by simply being very deliberate about how you decide to start and finish work!

This part of the book has an interesting twist. We connect *Little's Law* with *Flow Efficiency* but looking at it from a *TOC* perspective, while also exposing how it impacts the bottom line.

Part 2 - Acting on the Archimedean Lever that Boosts Performance: the Constraint

"Me. We."

Is said to be the shortest English language poem in history. It was given by Muhammad Ali talking to 2,000 Harvard seniors at a commencement ceremony in 1975. It expresses a sense of community, and of appreciation for support and togetherness.

The *Story of Herbie* and *Constraints Management* is evocative of this poem: if you want to improve and succeed, helping the weakest link in the chain is key.

And it has a nicer side effect: it will make you more profitable while moving closer to your Goal, whatever definition you give it!

Part 3 - Making Accounting Make Money

Causality is a very strong proposition. When you assert causality, you are making one of the strongest statements known to man. You are entering the scientific arena where there are no exceptions to the rules.

Causality and management are rarely effectively associated together. It is particularly demoralizing when, after reading this part of the book, one gains the insight and awareness that a great deal of the *operational* and *performance* problems that most companies experience, are actually rooted in the pervasive use of *Cost Accounting* to make critical decisions. Problems that are self-inflicted.

Decisions about business options or product/service mixes or prioritization of initiatives are often distorted and produce the opposite results of what is desired.

In this part of the book we will discover the virtues of *Throughput Accounting* - and the best part is that we do not need to be Certified Public Accountants nor have mastery of any higher levels of math to appreciate how it works and how it benefits the entire organization.

The ultimate impact of *Throughput* based decision making comes in the form of a cultural enlightenment, where all parts of an organization understand that they can work together as a team, rather than constantly quarreling about opposing priorities.

With *Throughput Accounting* all such conflicts give space to unanimous decision making. Naturally in companies where there is no infighting, more energy and trust can be dedicated to winning where it matters: winning the goodwill of customers and winning against competition in the marketplace.

Part 4 - Maximizing Business Value in Knowledge-Work

Agile and Kanban have very little to offer and fall short in dealing with today's complex environments.

First, they are totally oblivious to the true *Constraint* of the system. *Constraints Management* is the focus of this section, in particular in what we call PEST environments: where we have to confront multiple Projects or Products, Events (i.e. deadlines), Stakeholders and Teams.

A *Constraint* can appear anywhere in your *Work Flow*, *Work Process* or *Work Execution*. Distinguishing between them has enormous economic benefits; as is being able to establish if the *Constraint* is impacted by *Common Cause Variation* or by *Special Cause Variation*.

Detecting and managing *Common Cause Variation* is probably the single most important oversight in both modern management and Agile practices today. It is where we get a grip on systems to enable double loop learning, having a 360 degree view of the landscape at work. It is here that we can find out how to execute better, become more reliable, keep our promises and establish trust.

In this section, we will need to let go of another *Mental Model*: the pervasive use of *Column WIP Limits*! Paradoxically they are one of the biggest impediments to high performing flow and hinder the potency of the most effective WIP limiting mechanism discovered to date: *Drum-Buffer-Rope* scheduling.

At the end we will discover the best way to prioritize and sequence work to deliver the maximum business value for our company.

Part 5 - Preparing for High Performance Execution and Governance

Part five is pure *Tameflow Kanban*. *Flow Efficiency* and *Throughput* are cornerstones of the approach. How to visually instrument and operationally manage four different Kanban boards is covered: *Flow Efficiency* boards, *Drum-Buffer-Rope* boards, *Throughput* management Kanban boards and finally, Portfolio boards.

These boards impact culture via a reconception of common corporate social norms and values, the most prominent of which is the acclamation of idleness. It is not poetic license, but the deep insight that in high performing organizations, only the *Constraint* cannot afford to remain idle; while everyone else *should be idle at times*.

Of course, the key question is to know when non-Constraints should work and when they should not work; and also what else they should be doing during such periods of idleness.

Scope and minimalism. How do we wade through the internal and external forces that business imperatives and limited resources bestow upon us and despite all this, bring something to the table? For this specific purpose, the use of the *MOVE* - the Minimal Outcome-Value Effort - is of strategic value and favors a communion between upstream and downstream participants of the system. It is here that the arbitrage between units of stakeholder value, units of *Throughput*, units of costing and reporting and units of work and delivery are consolidated to create common motion and focus of effort.

Every project goes through its critical path during execution. It is here that Dr. Eliyahu Goldratt's execution management techniques (first introduced in his *Critical Chain Project Management*) based on *Buffer Signals* as leading indicators of risk materialisation become an extremely potent tool in handling what VUCA (**V**olatility, **U**ncertainty, **C**omplexity, **A**mbiguity), Mr Murphy and Chaos throw at us every day.

And it comes for free while it is at the root of the high performance *Management by Exception* practices promoted by the *TameFlow Approach*.

Another major practice introduced is *Full-Kitting*. Significantly, we examine how *Full-Kitting* needs to be considered in light of possible misconceptions originating from Agile thinking. With *Full-Kitting* we are setting up our work so that we can effectively realize the ideal conditions of "*One-Piece-Flow*," meaning that any piece of work will flow through the system with no interruption and with maximal *Flow Efficiency*.

Part 6 - Achieving High Performance Execution and Governance

Executives have a role to play as they act as the trigger of the chain reaction that will hit the system. How do we prioritize *MOVEs*? How do we rank them? How and when do we start work? How and when do we initiate *Flow*?

This is the operational aspect of *Full-Kitting* where executive management actively participates in a targeted and structured way in preparing for the execution of work entering the system, in particular by acting as tie-breakers when there are prioritization conflicts between different stakeholders.

What is so special about execution?

Well, it becomes a new dimension to manage. It gets very sophisticated and follows the lineage used throughout this book. The use of *Management Signals* helps us come to grips with the unfolding of all our projects so that we know how to make optimal operational decisions just in time when they are needed.

To conclude this part of the book, *Executive Governance* promotes a minimalist yet powerful management style which can cater for all issues and participants in full fairness. *Management by Exception* - based on leading *Management Signals* centered around the *Constraint* in the *Work Flow*, *Work Process* and *Work Execution* - is a unique and one of the most riveting innovations in knowledge work management uncovered in this book by applying the lessons of Dr. Goldratt and his *Theory of Constraints* to this domain.

It is a coherent and captivating ending to the book - with the *Constraint* as the central pivot of all our thinking and decision making.

Part 7 - Igniting High-Performance

Flow and effortlessness go together but it does not come naturally. It is a second nature that needs to be harnessed.

This last section of the book is about simple (proto-)patterns that will get you on the right path in guiding your first steps in the right direction.

They are so easy to understand that you could almost start reading them first!

Credits

Cover Image: *Fractal Phlegyas on the River Styx*, ©2008, Devin Moore

Typographical Conventions

Throughout this book, whenever you see capitalized words in italics, they represent some term that has a well defined meaning in the context of the *TameFlow Approach*. (Naturally italics are also used for emphasis and direct quotations where applicable.)

Any new terms are introduced with an icon like this:



New Term: the definition of the *New Term*.

This convention will allow us to distinguish between any word used in everyday terms, like constraint, as opposed to when it bears the specific meaning it has in the *TameFlow Approach*, like *Constraint*.

Cross-references to other chapters or headings will also appear in italics.

From time to time, we will encounter clarifying notes. Such notes will look like as follows:

Note: This is a note that further explains or refines the concepts described by the main text.

Disclaimers

Author Views

Views expressed in this book are those of the authors or the individuals quoted herein, and are not necessarily the views of TameFlow Consulting Limited, TameFlow Press or their respective associates and affiliates.

Quotations

Any quotations are made in good faith and reproduced on fair use basis, for the purpose of faithfully representing original thoughts and attributing their authors. Where the source is known, it is given. In addition to the Bibliography section at the end of the book, further sources are listed and updated at:

<https://tameflow.com/bibliography>.

If you are the copyright holder of any cited passage and feel you are being cited inappropriately, please contact us to clear the matter.

Testimonials

While we were circulating early drafts of this book, we received so many overwhelmingly positive testimonials from thought leaders, authors, professionals and practitioners in the fields of the Theory of Constraints, Throughput Accounting, the Kanban Method, Lean, Disciplined Agile, FLEX, Project Management (PMI), Agile and Scrum that it would take too much space to list them all here. Instead we decided to spread the testimonials throughout the book placing them at the end of the chapters, where most relevant.

Acknowledgements

The authors thank Carole Schultheiss for having proofread this book.

Special thanks for their extensive support to: Eli Schragenheim, Daniel S. Vacanti, Bob Sproull, Kirk Bryde, Jack Vinson and Ian Heptinstall who all held us to higher standards, challenged our thinking, exposed weaknesses and provided extremely valuable and constructive criticism.

Thanks to all other reviewers who gave us plenty of feedback: Al Shalloway, Christophe Achouiantz, Mario Latreille, Marcus Hammarberg, Sylvie Levesque, Curtis Hibbs, Stuart Burchill, Emiliano Sutil, Nicolas Desjardins, John Coleman, Paul Grenier, Minton Brooks, Daniel Gagnon, Tom Cagley, Karin Dames, Matthew Croker, Irmgard Barth, Daniel Plourde, Troy Magennis, Pepijn van de Vorst, Anna Sikorska, Michael Küsters, Wolfram Müller, Dimitar Bakardzhiev, Stefan Willuda, Kurt Häusler, Matt Barcomb, Corey Ladas, Robert Newbold, Henrik Mårtensson, Frode Odegard, Daniel Hernández, Stéphane Chouinard, Madeline Grejda, Katharine Chajka.

Dedication

To the memory of my father.

– Steve Tendon

À ma fille Magali.

– Daniel Doiron

Forewords

This book covers new ground, in particular by bridging two major schools of thought that are very relevant in today's knowledge-intensive organizations: the *Theory of Constraints* and the *Kanban Method*. In the quest to find some experts who could vouch for our effort, we were not able to find anyone with deep knowledge of *both* fields.

So we decided to ask *two* prominent experts in the respective domains to give us their opinion - and our call received a welcoming response from Eli Schragenheim and Daniel S. Vacanti.

Eli Schragenheim is one of the world's foremost experts on the *Theory of Constraints* and worked with Dr. Goldratt from the beginning.

Daniel S. Vacanti is likewise one of the world's foremost experts on *Kanban* and *Flow Metrics* as they apply to knowledge-work, and was also one of the very early proponents of the *Kanban Method*.

We are humbled and honored to have received their thoughts as you can read on the following pages.

Foreword by Eli Schragenheim

TameFlow is directed at knowledge-based organizations, or functions within large organizations, that are under constant pressure to deliver more and more answers to various requests. So, it covers a wide variety of organizations and functions.

TameFlow also challenges the mind of the reader. It raises various mental models, paradigms in my own terminology, which are widely used in such functions and put them under intellectual testing. I think this stretch of the mind is really useful. I truly believe that any person who uses either the traditional way of managing the flow of work or using the *Kanban Method*, should read the book in order to challenge himself/herself whether the pros and cons of the current way are logically clear.

To my mind the key need for the book is that it tackles a major universal problem causing many organizations to be trapped in a vicious cycle.

Many, probably most, large organizations, certainly corporations, find themselves constrained by their IT department, and the situation is getting worse with the growing flow of new technologies that seem to solve problems, but also cause huge damage. Almost all new technologies and innovative ideas depend on IT. So, organizations whose core capabilities and business are not in IT, are actually constrained by the capacity (sometimes also by lack of the appropriate capability) of their internal IT function. The continuation of a stream of new IT functionality, let it be the Cloud techniques, Artificial Technology (AI), Business Intelligence (BI), cyber protection or massive digitization (Industry 4.0) puts unbearable burden on the IT department to the extent that its performance becomes chaotic. Keep in mind that every IT department has to keep on supporting the users with all the current software and hardware requirements.

I suggest that every organization has to manage at least two critical flows. The first one, **The Flow of Value**, includes all the parts of the way the organization delivers value to its customers.

The second flow, **The Flow of Initiatives**, consists of all efforts to improve the *Flow of Value* in the future. When the IT is a bottleneck it blocks mainly the *Flow of Initiatives*. It endangers the competitive advantage of the company in the future. All top management of banks would agree to this assertion. All manufacturing organizations are facing a flood of new software functionalities to support Industry 4.0. How much of those can the current IT department implement successfully while continuing to manage all the ongoing requests?

TameFlow offers a combined approach of the *Theory of Constraints* (TOC) and the *Kanban Method*, and it challenges the mental models of the reader along the way. When this is directed at **the constraint for the growth of the entire organization** I rationally conclude that reading the book is a very beneficial start to deal with the source of the pain.

— **Eli Schragenheim**, CEO of Elyakim Management Systems, author of “*Throughput Economics, Making Good Management Decisions*,” “*Manufacturing at Warp Speed: Optimizing Supply Chain Financial Performance*” and “*Management Dilemmas: The Theory of Constraints Approach to Problem Identification and Solutions*.”

Foreword by Daniel S. Vacanti

Most practitioners think *Kanban* is all about finding and fixing bottlenecks. That's not true.

The above sentence may seem strange given that I am writing a foreword to a book that is about - amongst other things - applying the *Theory of Constraints* (TOC) to flow. But bear with me and I think we will get this all sorted out.

To understand the misconception around *Kanban* and *TOC*, we must go back to Kanban's very beginnings (at least for knowledge-work, anyway). My intention is not to waste your time rehashing the origins of Kanban - others have done that better than me (specifically, I refer you to Darren Davis and his excellent video and blog, "*The Secret History of Kanban and Why It Matters*"). What is important to emphasize, however, is that from its very beginning, *Kanban* has always been about an attempt to achieve flow.

"But," you may ask, "isn't removing bottlenecks what achieving flow is all about?" In a word, no. The problem is that every time someone in the *Kanban* community hears the word "bottleneck", they want to rush in and immediately apply *TOC*. This misapplication of *TOC* is why Steve and Daniel's book is so important. By reading this book, what you will learn (amongst many other things) is that the application of *TOC* to a bottleneck that is not the true system constraint will usually only serve to *make things worse*.

This is why the management of flow is so tricky. What looks like genuine, constrained flow (i.e., "bottleneck") may in fact only be a transient condition caused by some other process variability. Applying *TOC* in this case would be much like using a shotgun as a fly swatter.

Unfortunately, this misunderstanding has permeated and persisted within the *Kanban* community since its very beginning. This book fixes all of that.

Please do not mistake me: Steve and Daniel's work is not just a recipe for how to apply *TOC* to achieve flow. Rather (and this is what I love about what they have done here), it is a thought-provoking discussion on how to deeply examine your process to identify the opportunities for improvement that will provide the most bang for the buck. Visualization, though important, is not enough. Limiting WIP, though important, is not enough. Only a deep understanding of the principles of flow and the factors that affect flow will yield true, overall process improvement. And, by the way, if you were assuming that this stuff only applies to work being done at the lowest team level, then think again. Improvement must be sought at all levels of the organization (team, project, portfolio, etc.) and across all dimensions of flow (work, knowledge, value, etc.)

There are dangers lurking out there that we must be mindful of: flow is unintuitive; much of what is talked about in the Lean-Agile circles concerning flow is simply wrong. These reasons are why this book is such a critical addition to the *Kanban* literature.

I do not pretend to understand all of what Steve and Daniel talk about, but I am sure glad that they have invited me along for the ride. You should be too.

— **Daniel S. Vacanti**, CEO, ActionableAgile, author of "*Actionable Agile Metrics for Predictability, An Introduction*" and "*When Will It Be Done? Lean-Agile Forecasting To Answer Your Customers' Most Important Question.*"

Introduction

Introduction by Steve

This book will present novel ways to manage knowledge-work at scale by taking inspiration from and further expanding the application of the *Theory of Constraints* of Dr. E. Goldratt, author of the famous business novel “*The Goal*.“

The ideas presented in this book have been developed and proven in the field since the beginning of the 2000s, and are now collectively known as the *TameFlow Approach*. I defined the *TameFlow Approach* in a more systematic way with the first self-published book of 2013 “*Tame the Flow*” which then turned into a revised edition by J. Ross Publishing with the title of “*Hyper-Productive Knowledge Work Performance: The TameFlow Approach and Its Application to Scrum and Kanban*” in 2015 - which, by the way, is a must read if you want to have the full context of this work.

The *TameFlow Approach* has been successfully applied in a number of situations. From small start-ups to major multinational corporations.

The crowning masterpiece case of the *TameFlow Approach* was with a leading automotive component manufacturer, serving all major car brands globally.

It had a critical business unit with 8,000 people in 120 teams spread over a number of countries, across all continents, with cultural and coordination problems. The components were developed by a diversity of engineering specialties, including: mechanical, chemical, electrical, electronic and software engineering and other specialized groups. Any single change request could involve the work of any combination of the 120 teams. There were functional and temporal dependencies between the teams. Some activities were seasonal, and could only be performed during certain periods of the year. Naturally, there were deadlines (trade shows, contractual terms).

They faced the challenge to tackle 70,000 (seventy thousand!) “change requests” per month.

And there were always some change request that was “more urgent” than others.

The assignment was to make sense of the complexity and to be able to prioritize all change requests, all the while coordinating all teams, synchronizing their efforts and keeping the maximum number of clients happy.

In this scenario, conventional project management approaches had proven incapable to provide any workable solutions. Attempts at adopting Scrum and Kanban to make the teams more agile, did not produce any major benefits.

What could be done?

Developing a deeper understanding about how the *Work Loads* facing the organization could be rethought in terms of *Work Flows* and *Work Processes*; focusing on the *Constraint* therein; and being in charge of unpredictable factors that happen during *Work Execution* turned out to be the resolute approach.

This book will introduce you to the fundamental ideas of *Operational Flow* management and *Constraints Management*. It continues where conventional approaches fall short.

Focus is squarely on producing tangible bottom line results for businesses, all the while addressing the issues of coordination, synchronization and prioritization of multiple projects, with multiple stakeholders, employing multiple teams.

At first you will discover how to gather the low hanging fruits: how to produce outstanding bottom line results without the need to invest, reskill, retrain, restructure or take upon any new unknown risks.

Then you will learn how to increase organizational performance by learning *exactly* where to focus investments and actual operational improvement initiatives. Finally you will acquire an understanding about how to coordinate, synchronize and prioritize multiple *projects*, with multiple *events* (or deadlines) and *stakeholders* across multiple *teams*.

You will learn about the *TameFlow Approach*, and how it can be used to improve organizational *Flow* and show the way for your business to improve on all fronts it is engaging.

– **Steve Tendon**, MSc Software Project Management, Inventor of the *TameFlow Approach*

Introduction by Daniel

In the economy of the 20's, we need to think differently and show results.

Since the beginning, Agile (Scrum, SAFe, Scrum@Scale, Nexus, Less), has given too much importance to approaches that have yet to find the path to significantly improving the earnings section of financial statements.

Mainstream Kanban has forgotten the teachings of Dr. Goldratt on the *Theory of Constraints* (TOC) and in particular *Throughput Accounting* (TA). Where do Agile practitioners turn for leadership when time to market is not the imperative but generating more sales (Throughput) is? This is a legitimate business issue that goes unanswered.

This book gives you actionable knowledge and an advanced discussion of *Tameflow Approach*, an approach to knowledge-work management that focuses on Throughput and Flow Efficiency.

This book came unexpectedly to both Steve and me, and still surprises me today by its depth. It will have an impact on how we all envision knowledge-work forging ahead.

— **Daniel Doiron**, CTT, CKP, Certified Tameflow Trainer

Prologue

The Story of Herbie

This is a story about a team of scouts who set out for an overnight hike. They start their walk early in the morning, planning to hike through a forest to a remote place where they will set up camp, sleep overnight and then walk back the next morning.

The trail in front of the team can be imagined as a long series of steps, like this:



The walk starts briskly. The terrain is unchallenging and the trail is well marked and easy to follow.



All scouts carry their own rucksacks.

The terrain becomes more difficult, with undergrowth on either side forcing the hikers into a single line.



As the terrain becomes more and more difficult, the line stretches longer and longer. The troop leader is concerned about keeping the team together to ensure that nobody gets lost in the forest.

One of the scouts is a little overweight.



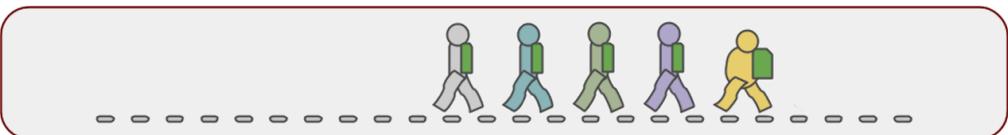
He also carries a backpack that is overloaded.



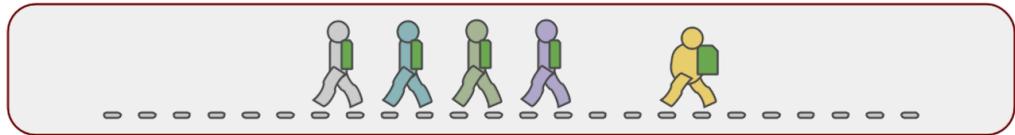
At first, the overweight scout can keep pace with the line. But he is slower than the others and, in effect, is holding up those behind him, who eventually overtake his position.



The troop leader encourages the team to close ranks, and keep together - so, one after the other, all those behind the slow scout overtake him. Eventually he drops to the end of the line.

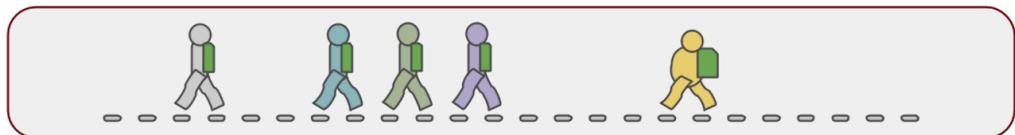


He is still slower than everybody else. Eventually a gap develops and he walks alone, far behind all the others.



That scout is Herbie. The troop leader calls him: "**HERBIEEEE!**"

Now the leader is even more concerned the boys will get lost in the woods. Not only is Herbie being left behind, but the faster ones are pulling away in front.



The troop leader shouts "**C'MON, HERBIE! SPEED UP,**" to encourage the scout to walk faster; but he's already walking at the maximum of his capacity.

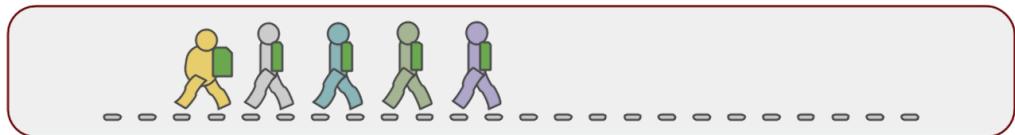
Eventually the leader understands that Herbie's speed is determining the troop's ability as a whole to move faster; but Herbie can only walk at his own pace, slower than everyone else. Herbie is determining the troop's "*Throughput*" in terms of how much terrain they actually cover per unit of time.

Note: In normal conversation we would refer to this as the "*speed*" of the team - but we introduce the term "*Throughput*" here because as you will see it will be occurring throughout the book. *Throughput* is the amount of work or value delivered per unit of time.

The troop leader realizes that whoever is moving the slowest in the troop is actually determining the overall *Throughput*. That role can actually float from one scout to another: it depends on who is the slowest at that moment in time.

At this moment, Herbie has the least capacity for walking; and his speed ultimately determines the speed of the entire troop.

The troop leader then puts Herbie in *front* of the line and commands: "**Everybody stay behind Herbie!**" and also gives instructions that nobody should pass anyone, but rather, keep to his position.

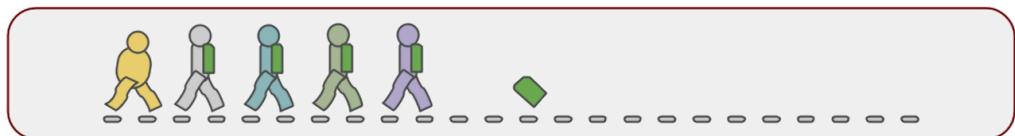


In that way, the team can stay together and nobody is out of breath. Any gap that develops can easily be recovered, since everybody can walk *faster* than Herbie who is at the front. Now at least, nobody will get lost in the woods.

However, time is running out and they run the risk of not reaching camp before dark. The troop must walk faster.

Herbie is not only overweight but his backpack is also overloaded. His backpack contains a six-pack of soda, a few cans of spaghetti, a box of candy bars, a jar of pickles and two cans of tuna fish. In addition, for the sake of “*being prepared*” (as a good boy scout) he has also packed a raincoat, rubber boots, a bag of tent stakes, a large iron skillet, and a collapsible steel shovel.

The troop leader realizes that if some of the load could be taken off Herbie, Herbie would be able to walk faster. So he commands: “***Everybody carries a piece of Herbie’s gear.***”



With the new arrangement, the team can now walk at double the speed and they are still able to keep together behind Herbie. They arrive at their night camp late in the afternoon, just before dusk. The troop leader reflects on the fact that Herbie was key in governing the performance of the *entire* team.

Herbie and Your Work Flow

The troop of hiking scouts is similar to a system that has to produce a product or deliver a service. The troop delivers “walked trail” - that is, the distance that the whole team has covered together. The head of the line begins work by “processing” the trail in front of the troop, which is the equivalent of raw material, or work to be done.

When the first scout in line walks, he “processes work” and hands it over to the second one in line, followed by the next, and so on until the last one. Each scout represents an operation or a step in a production or service delivery process. No matter the order in which the scouts are lined up, they represent a series of *dependent* events, subject to *statistical fluctuations* on how fast they can walk. Unfavorable statistical fluctuations will *accumulate* rather than average out which explains why the distance between the front and the back of the line continuously increases. Of course, no matter what, the one *Critical Success Factor* that must be met, is that they all arrive safely together at the base camp.

Note: The individual scouts might appear as *independent* processes. However, note that the story line makes them walk on a singletrain - they are forced to walk exactly *the same path*. What is walked by the first one, must be walked by the second, the third, and all of them till the last one. If they were taking different paths to basecamp it would be independent. In virtue of this, we can observe the impact of variability (i.e. statistical fluctuations) on dependent processes. Delays just keep on accumulating and never “average out.”

Not until the last scout has walked a step, can that step be considered as delivered.

All the steps between the first and last scout is the “inventory” or *Work in Process*. When the distance between the first and last scout increases, there is more *Work in Process*. The *Throughput* of the system of the walking scouts corresponds to the rate of their walking *together* as a team (their speed).

Operational Expense is the energy required by the scouts to sustain the walk. Any time they hurry to close a gap, *Operational Expense* increases because more energy is consumed.

The Five Focusing Steps

The way in which the troop leader rearranged the order of the line and redistributed the load to carry is an example of the so called *Five Focusing Steps* (5FS) of the *Theory of Constraints* (TOC). The 5FS are the foundation of the process of ongoing improvement that characterizes the TOC.

The 5FS are stated more clearly as follows:

1. Identify the *Constraint*
2. Exploit the *Constraint*
3. Subordinate to the *Constraint*
4. Elevate the *Constraint*
5. Repeat

Note: At times the original language used in the *Theory of Constraints* might not seem very considerate to human needs, and some react negatively just because words like “exploit” and “subordinate” are used. If one prefers, there is also the 5FS acronym **FOCUS** by David V Hodes: (1) **F**ind the *Constraint*; (2) **O**ptimize the *Constraint*; (3) **C**ollaborate around the *Constraint*; (4) **U**pshift the *Constraint*; and (5) **S**tart again! - In any case, no matter what choice of language one does, what matters are the underlying concepts and the thinking behind the 5FS.

Let's look at the 5FS in more detail.

Step 1: *Identify the Constraint - “Herbieeeee!”*

A production/delivery system is considered as a chain; and the chain has a weakest link. That weakest link is the *Constraint* that prevents the entire system from achieving better performance. The first step is to “*identify*” the *Constraint*, so one knows exactly where to intervene. In the example, the weakest link was Herbie.

Step 2: *Exploit the Constraint - “C'mon Herbie! Speed up!”*

The second step is to “*exploit*” the *Constraint*, in the sense that you must ensure that it is working to the maximum of its (sustainable) capacity. This is what the leader was doing when he kept on encouraging Herbie to walk as fast as he could.

Step 3: *Subordinate to the Constraint - "Everybody stays behind Herbie!"*

The third step is to ensure that all other resources of the system “*subordinate*” to the *Constraint*. By putting everybody behind Herbie, the leader made sure that the line was kept short; that nobody ran away in front; and that they all arrived together at their destination. By means of subordination, overproduction is avoided, while inventory and work in process is kept under control.

Step 4: *Elevate the Constraint - "Everybody carries a piece of Herbie's gear!"*

The fourth step intends to “*elevate*” the capacity of the *Constraint*. By elevating the capacity of the *Constraint*, the entire system will perform better; it will perform at the new level reached by the *Constraint*. In the story, Herbie’s walking speed increased once items were removed from his backpack. As a result, the entire team could progress faster along the trail.

Step 5: *Repeat!*

While not illustrated in the story, the final step is to go back to step one. In particular, this should happen if the elevation of the *Constraint* actually “*breaks*” the *Constraint*. In the above example, this could happen if Herbie, with a lighter backpack, could walk faster than someone else. In that case, some other scout would become the new *Constraint* of the team. Any time a *Constraint* is elevated, it is possible that it will no longer be the *Constraint*. Some other person (“resource”) might take its place. Also note that a *Constraint* may move because of other events, unrelated to Herbie himself. For example, another scout might have developed a blister on his heel which is now slowing him down so much that he cannot keep up with the others, including Herbie.

The fifth step is also accompanied by a warning, suggesting you should not allow “*inertia*” to become a *Constraint*. At times, complacency and relaxation take over and you are no longer actively looking for the real *Constraint*.

The Unstated Step 0

Often a “*Step Zero*” is also listed: that of determining and agreeing on the “*goal*” that the system needs to pursue. Without agreement on the common goal, it is impossible to determine if any change is an improvement. In the case of the scouts, the goal was reaching the night camp before dark.

The Extra Step 6

On a more strategic level, Step 6 is considered whereby you deliberately redesign your system and decide where you want to place the *Constraint*. By intentionally deciding where to place and where to maintain the *Constraint*, all other subordination and elevation decisions become easier. By keeping the *Constraint* in a given place, the whole system becomes simpler to manage because it remains stable. This is very important, because (as we will see) stability is also a prerequisite in order to apply the ideas of the *TameFlow Approach*.

The Secret of all Steps

The scouts’ performance became possible only because they had a common goal (getting to the base camp before sunset) and because Herbie trusted his peers to carry his gear. The story of Herbie and the

scouts embodies the noble patterns of *Unity of Purpose* and *Community of Trust*.

The troop leader, with his evolving insights and updated mental models of what was happening during the hike, embodies the pattern of *Inspired Leadership*. The leader was *inspired* by thoughts and ideas to act in order to improve the situation of his team.

Note: It is important to recognize the choice of words here: it is about “*inspired*” and not “*inspiring*” leadership (the latter being much more commonly used in the management literature). The sense is that the leader begets new ways of seeing reality (through new *Mental Models*) to the extent that the new understanding sets off - the leader gets “*inspired*” - novel decisions and actions.

The patterns of *Inspired Leadership*, *Unity of Purpose* and *Community of Trust* are at the center of the *TameFlow Approach*. Notice that the ideas, though, came from *observing* and *paying attention* to what was happening with the team, at every moment.

Anything we will learn about the *TameFlow Approach* will either be supportive of or derived from these three noble patterns.

We have Herbie to thank for them.

Note: This story was reproduced and adapted from *Chapter 12 - Herbie and Kanban* of the *Hyper-Productive Knowledge Work Performance* book by Steve Tendon and Wolfram Mueller, J. Ross Publishing, 2015. The original story of Herbie was first described by Dr. Eliyahu Goldratt in his business novel *The Goal* (North River Press, 1992).

Daniel Gagnon says...

Co-authors Steve Tendon and Daniel Doiron have just released what to my mind is possibly the most thought-provoking, assumption-challenging book about knowledge-work in general and agile in particular since Don Reinertsen's 'Flow'.

*In this work, which is in some ways a streamlined and updated version of the ideas Steve and co-author Wolfram Müller first introduced us to in their 2015 release 'Hyper-Productive Knowledge Work Performance', Steve and Daniel make a compelling case for uplifting the Kanban Method with practices and tools from the Theory of Constraints - and in the process bring back Herbie, the unforgettable character from Eli Goldratt's 1984 *The Goal*.*

A quicker and lighter read than the (excellent) 2015 offering, 'Tame your Workflow' still hits all the right notes and will leave Agilists of all stripes - as well as anyone with an interest in knowledge-work - with a sense that here at last is a higher order of thinking, one that goes beyond the usual how-many-angels-can-dance-on-the-head-of-a-pin type of casuistry and semantic diffusion to which agile seems to have mostly fallen prey.

Read it to challenge yourself and perhaps take away some applicable ideas - there are many here.

Daniel Gagnon, Disciplined Agile Fellow and Organizational Agility Advisor.

PART 1—Getting More for Nothing with Flow Efficiency

Superior organizational performance in knowledge-work is all about decision-making. It is both about deciding to do the *right thing*, and to do the *thing right*, as a common meme suggests. In a setting of equivalence of decisions, speed of decision-making and acting upon those decisions will make all the difference.

Making the right decision at the right time becomes paramount. No matter what decisions we undertake, they are always grounded in our understanding of the world. We resort to our *Mental Models* to navigate the complexities of every day business, and get their guidance on what are the *right* decisions.

Often we act to the best of our ability and experience, but the *Mental Models* supporting our decision-making might not be adequate. We need to be open minded, and considerate of new or alternative *Mental Models* that could dramatically change the way we understand the world, and thus how we make decisions.

In this first part of the book we will examine a few *Mental Models* in relation to the very concrete problem of improving *Operational Flow*. We will see how *Mental Models* can have a dramatic impact on our performance, and often without having to sustain any major effort or investment.

Let's discover the power of *Mental Models*.

John Coleman says...

It can be too easy to dismiss the Theory of Constraints for knowledge work and there is a danger in doing so. It comes closer to home when stable & ridiculously expensive & constrained physical things begin to impede our knowledge-work. And constraints don't need to be physical; sometimes there are limits to growth.

Throughput Accounting also changes the game.

TameFlow is Kanban for grown-ups, for teams, for teams of teams, for organizations. It's not just about optimizing flow; it's about optimizing throughput in the pursuit of positive impact.

"Tame your Work Flow" has got to be a top 10 read for any self-respecting 20's knowledge-worker.

John Coleman, independent agility coach, trainer & strategist.

1—The Power of Explicit Mental Models



What is this *TameFlow*, anyway?

Did you notice the cover image of the book, reproduced above. There's a story to be told. In the "Inferno" of the *Divine Comedy* by Dante Alighieri (1265 -1321), Phlegyas was the only daemon with the power to cross the river Styx, the river at the boundary between Earth and the Underworld. With this power, Phlegyas would ferry the souls of the dead to the Underworld. In the picture you can see the old bearded man waving an oar over the waves of the river.

The man is taming the flow of that furious river!

If that sounds like running a business in modern times, it probably is so.

The above story superbly represents the idea of *TameFlow*! Like Phlegyas taming the flow of waves on the river Styx, *TameFlow* will tame the flow of work, money, information and even well-being throughout your organization.

TameFlow has many facets. It is rooted in the *Theory of Constraints* of Dr. Goldratt; it is based on *Pattern Theory* as taught by Christopher Alexander; it has been influenced by many schools of thoughts about management and organizational design.

More than anything else *TameFlow* is founded on *Mental Models*, which give individuals and organizations new perspectives through which they can reflect on what it is they do to achieve their goals. *TameFlow* appeals to team members as well as to C-level executives, and any one in between. It benefits external parties, stakeholders, shareholders, suppliers and of course, customers.

TameFlow has been forged and tested in the field. It has been successfully employed by small startup teams and at scale by multinational corporations - and everything in between. *TameFlow* even determined the competitive strategy of an entire country, as it was instrumental in creating the *National Blockchain Strategy* of the Republic of Malta – which other approaches can claim to have influenced the economy of an entire nation?

TameFlow changes and improves the way businesses pursue bottom line results, in measurable ways: CFOs love it, because they discover the connection between “business agility” and their accounts. HR and employees love *TameFlow* for the attention given to human and psychological factors. It brings entire organizations to a state of hyper-performance, thus keeping everyone happy at work. Even customers love its effects, because they receive more value, sooner.

What is *Flow* and *Throughput*?

TameFlow derives its name because it focuses on *Flow*, and like Phlegyas taming the forces of the river Styx, *TameFlow* tames four *Flows* that are at work in any organization. In particular, *TameFlow* deals with:

1. Operational Flow
2. Financial Flow
3. Informational Flow
4. Psychological Flow

Let us see what they are about in more detail:



Operational Flow: *How well are we delivering?* The *Operational Flow* is the conventional “*workflow*” that determines how work moves through an organization.

Raw materials are transformed into finished goods. Services are delivered through a succession of steps. *Operational Flow* is measured by means of *Operational Throughput*.



Operational Throughput: the rate (or “speed”) at which goods or services are delivered to the organisation’s customers.

TameFlow will produce substantial improvements in a business’s *Operational Flow*, by increasing *Flow Efficiencies*, reducing *Flow Times* and increasing *Operational Throughput*, as we will learn throughout this book.



Financial Flow: *How much wealth are we creating?* *Financial Flow* is what determines how money is generated by the organization.

Financial Flow is measured in terms of *Financial Throughput*.



Financial Throughput: the rate (or “speed”) at which an organization turns its products or services into profit (or other units of value).

Most organizations’ financial strategies are driven by cost saving measures. Instead, *TameFlow* will train managers to think in terms of *Financial Throughput*, and give them new insights which will radically reshape their strategies for making money. By focusing on the rate of value generation, rather than on cost savings or on the magnitude of profit alone, the organization’s financial performance will increase significantly.



Informational Flow: How well are you communicating?

Information is the lifeblood of any modern organization; even more so in knowledge-based organizations. It has a deep impact on the organization’s performance, building its collective intelligence and setting its ability to adapt and grow.

TameFlow will improve *Informational Flow* by establishing effective *Patterns of Communication* and *Interaction* as well as refining the organization’s effective design and structure to better support the flow of critical information.

The effectiveness of *Informational Flow* is determined by the right information reaching the right people at the right time.



Psychological Flow: How happy are your people? What can you do to nurture a humane, inclusive and gratifying culture, that caters for the needs of everyone? Such is the focus of *Psychological Flow*.

The highest levels of individual or group performance are achieved when people reach heightened focused mental *Flow States*, which is generally associated with a state of happiness, as studied by the Hungarian-American psychologist Mihaly Csikszentmihalyi.

In such mental states there is seamless concurrence of thoughts and actions. People experience complete control over their situation, even in the face of high risk, extreme difficulties and seemingly insurmountable challenges. *TameFlow* activates psychological flow in both individuals and groups by establishing “*flow triggers*” and the best environmental conditions to cultivate mental *Flow States*.

The clout of *Psychological Flow* unmistakably manifests itself by all people becoming *happier* at work.

Decision-Making through Explicit Mental Models

In today’s world of information overload, we are all forced to make a number of decisions on a daily basis. Speed of decision making is often a critical element. Whether deliberate or not, we often use shortcuts to help us decide.

One example is the use of *Mental Models*. (See P. Senge’s book “*The Fifth Discipline*”).



Mental Model: “An explanation of someone’s thought process about how something works in the real world. It is a representation of the surrounding world, the relationships between its various parts and a person’s intuitive perception about his or her own acts and their consequences. Mental models can help shape behaviour and set an approach to solving problems (similar to a personal algorithm) and doing tasks. A mental model is a kind of internal symbol or representation of external reality, hypothesized to play a major role in cognition, reasoning and decision-making.”

Source: [Wikipedia page on Mental Models](#)

Often, when facing a decision, we make it without any deeper thinking. We just instinctively know what is the “right thing to do,” supported by our beliefs and experiences. We are making such decisions on the basis of some *Mental Model*; on some well held beliefs in how the world works.

Mental Models are very useful. They may be imperfect; but they are practical. They provide a lens through which we can interpret reality and thus avoid overthinking and overanalyzing. We can then rapidly reach a decision and decide on a course of action. *Mental Models* help us filter through immense amounts of data and information, without becoming overwhelmed.

Typically, *Mental Models* are tacit and are assumed to be understood the same way or shared by others. Miscommunication is often the result of people not sharing the same *Mental Models*.

Most *Mental Models* are also flawed, because:

1. *Mental Models* offer an *imperfect* interpretation of reality. Hence, one strategy to improve the quality of our decision making, is to explicitly focus on better, context-appropriate *Mental Models* and on making sure that they are well articulated, shared and used by all people in our organization to frame whatever their object of attention might be.
2. *Mental Models* are often based on *wrong assumptions*. Once such assumptions are exposed and reconsidered, then new, more powerful *Mental Models* can be derived. They are often bewilderingly simple, to the point that they will seem *obvious*.

Oftentimes, the new *Mental Models* are indeed *obvious* in hindsight; but that does not imply that they are *self-evident* to begin with. Generally, we need to do some *deep thinking* and question our own assumptions before the *obviousness* is uncovered and becomes *self-evident*.

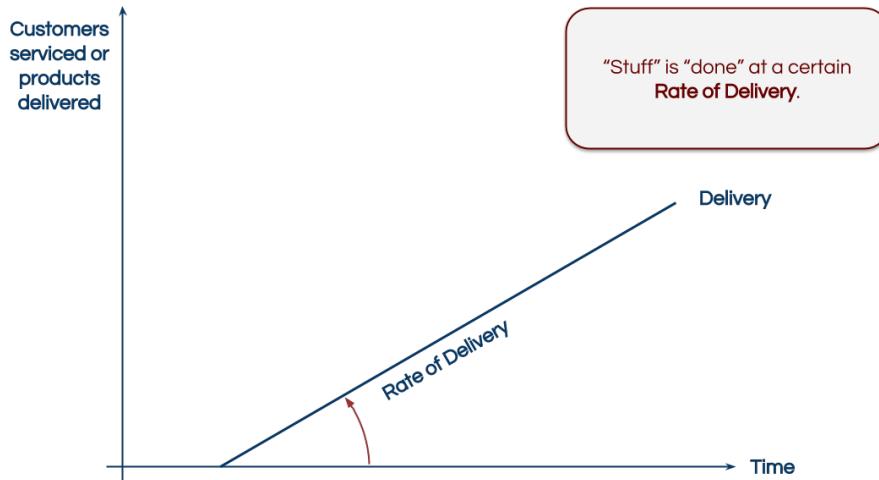
Mental Models have a profound impact on how we make decisions and hence, on the overall performance of our businesses.

A Mental Model to Explain the Business Value of Flow

To illustrate the power of *Mental Models*, let us try to understand why it makes business sense to focus on *flow* by highlighting the underlying mental wiring.

Note: While we are primarily concerned about *Operational Flow* and *Financial Flow* in this book, let’s not forget that the *TameFlow Approach* also considers *Informational Flow* and *Psychological Flow*, however the latter two are not the main areas of focus for this book.

Any business is concerned with how much work, value or financial results it is able to produce as time goes by. Typically, the situation is represented by a simple graph that shows how “well” the company is doing over time. It might look like this:

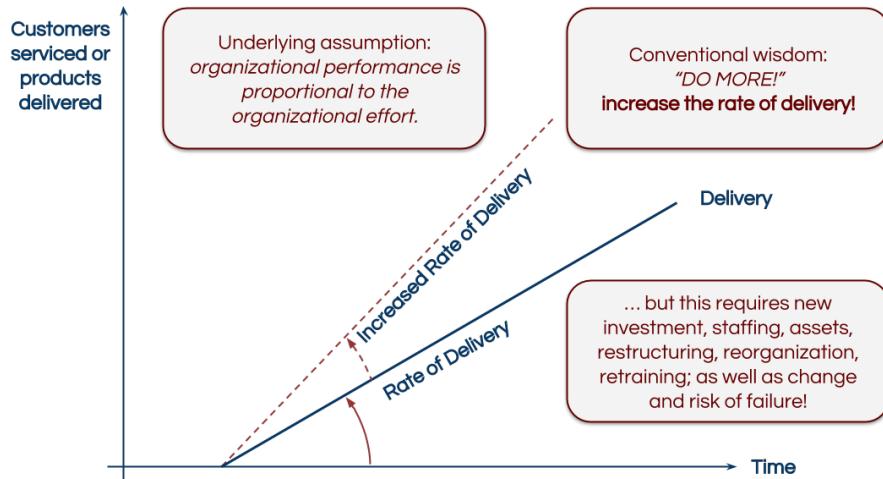


Of course, this is a simplification – it is a model, after all. One such simplifying assumption here is that the rate of delivery is linear. The purpose is not to have a perfect model of reality, but to reason about it and see if we can gain some deeper insights.

The Desire to Increase Performance

Most organizations are under pressure to constantly *deliver more*. Typically this happens because of the competitive market landscape in which they operate.

It is very common for business owners to *push* the organization to deliver more. Improvement initiatives might be undertaken, with the intent to increase the rate of delivery. Such desires can be visualized like this:



One common assumption is that the organization's performance is directly proportional to the organizational *effort*. Hence, the belief that if only the organization would put in "*more effort*," it would be able to produce more work per unit of time, and the rate of delivery would increase.

Increasing Performance through Effort Bears Many Downsides

However, increasing the organization's effort, typically bears costs, like:

- Investments (e.g. purchasing new equipment)
- Staffing (e.g. hiring new people and training them)
- Assets (e.g. buying new offices)
- Restructuring (e.g. putting new management structures in place)
- Reorganization (e.g. moving to new premises, changing lines of reporting and communication)
- Retraining (e.g. getting current staff up to speed on new mental models, concepts and practices)
- ... and so on!

The Rate of *Demand* is Important Too

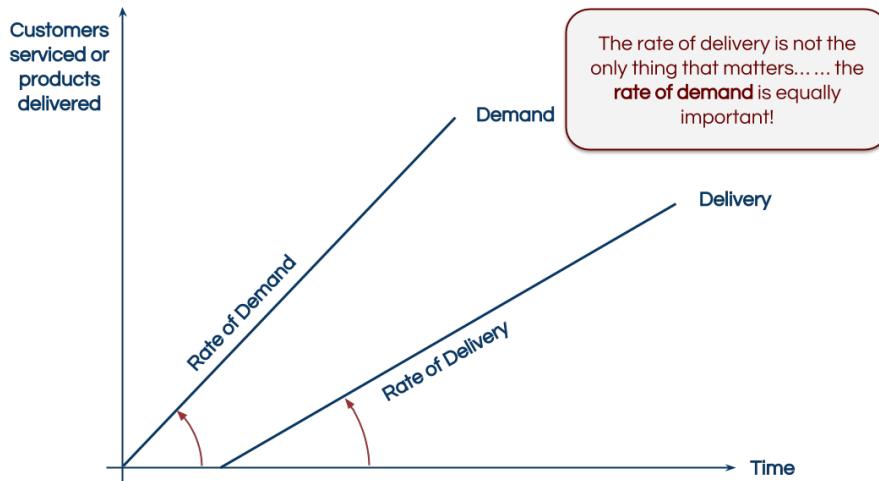
Most improvement operations focus only on trying to increase the slope of that delivery line.

Note: Here and in the illustrations of this chapter we refer to the *Demand* and the *Delivery* line. Of course it must be understood that these lines are a generic representation of the *Work* “to be done,” and is measured in the organization’s own chosen units like *Work Items* or *Projects* or *Tasks* that are moved in the *Work Flow*.

Another, often unstated, assumption is that the desired increase in the rate of delivery can cope with and match the rate of demand; but *that rate of demand* is often not taken into consideration at all.

If the rate of demand were taken into consideration, we would have to realize that demand often arrives at a higher rate than the rate of delivery; and that it happens *earlier* in time (because work needs to be done between receiving the demand and delivering the work).

It might be represented like this:



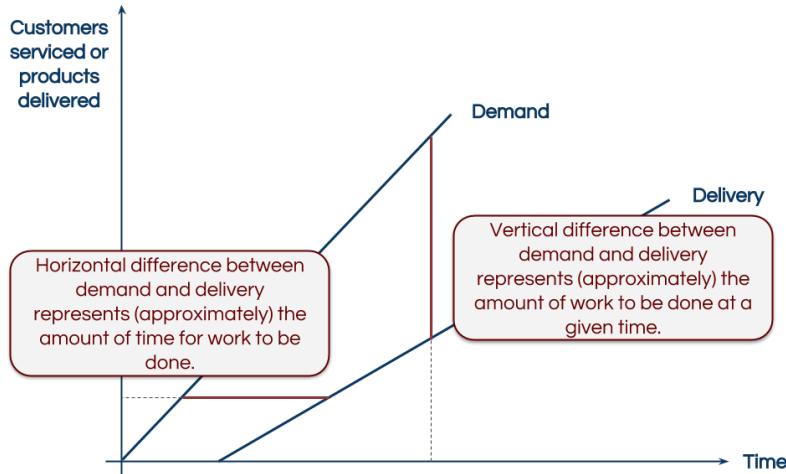
Now that we have plotted these two fundamental lines, the demand line and the delivery line, we can start reasoning about the “economic geometry” of the model.

Note: These two lines are fundamental in the so-called *Cumulative Flow Diagrams* (CFDs), which are well known tools for managing flow. However, here we are not concerned with the finer details of such diagrams, but only with the conceptual model we are developing.

The Difference Between Demand and Delivery is Even More Important

If we consider the difference between the demand and the delivery lines, then the vertical distance between the two lines represents how much work needs to be done at any given moment in time. This distance represents the work piled up inside the process or **Work in Process** (*WIP* for short). The horizontal difference represents how long any given piece of work will take from the moment the demand is created, to the moment the work is delivered.

We can illustrate the idea like this:



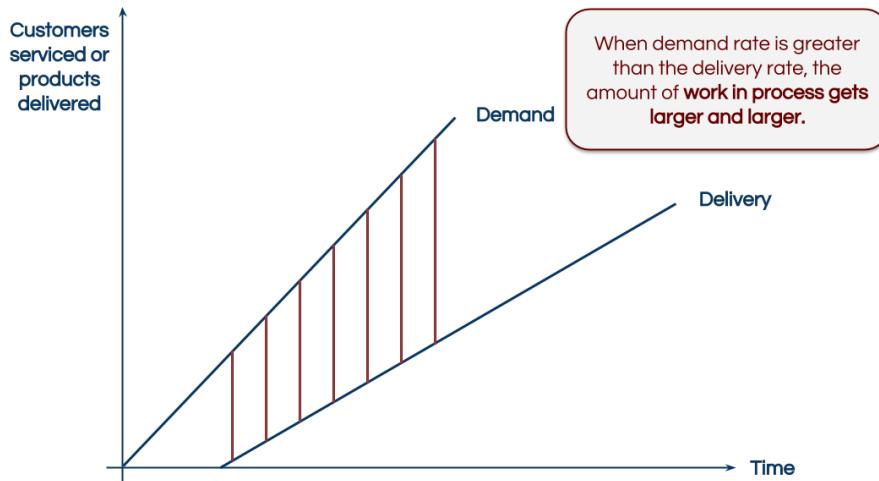
Note: An important clarification is necessary here. We must stress again that these diagrams are just a *model* in support of our reasoning and decision making. If we were to plot the actual *Cumulative Flow Diagram* of any system, then the difference between the two lines would not represent the *exact* amount of work and time in process; instead they would represent the *approximate average* of work and time in the process. To understand the details of this, see Daniel Vacanti's *Actionable Agile Metrics for Predictability* book.

Why Companies are Always Overburdened

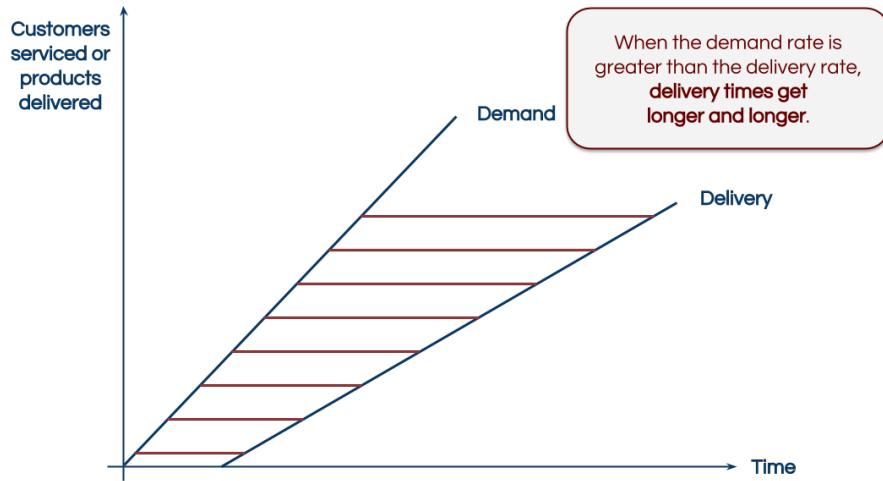
It is very common for most organizations to feel stretched, as if they were in chronic overload mode.

When the arrival rate of demand is greater than the production rate of delivery, it is not surprising. Have you ever had the feeling that work just piles up while it takes longer and longer to finish things? If so, then the following two diagrams can explain why.

The first diagram below shows how **the amount of work keeps on increasing** as time goes by.



And this diagram below shows how **the amount of time to finish the work just gets longer and longer**.



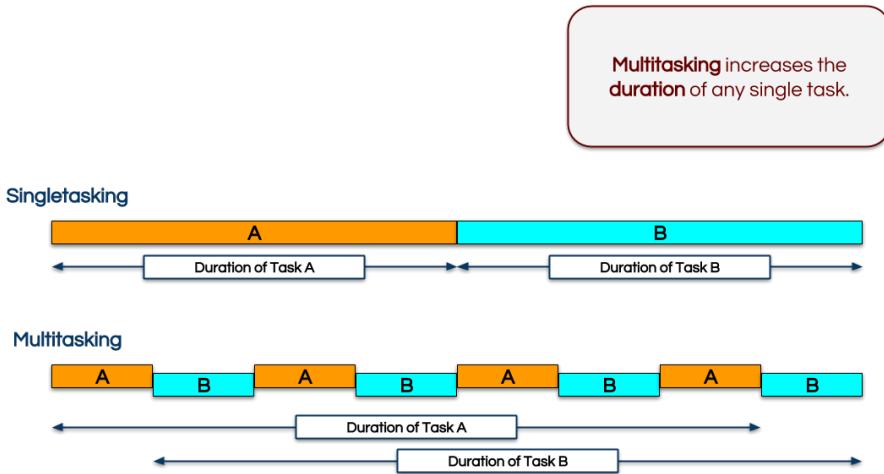
The Hideous Effects of Multitasking

The reason why work actually piles up is not only due to the continuous arrival of new demand, which arrives faster than it can be handled. It is also a direct consequence of the fact that (in most cases) **the decision is made to start new work as soon as it is received**.

This inevitably produces *Multitasking*.



Multitasking: “Stopping work on a task before it is completed in order to start work on another task.” (Source: TOICO Dictionary, Sullivan, 2012.)



It is clear, if we have to share our time between two tasks (of the same size), both of them will take twice as long compared to doing any one of them alone. If we have three tasks, they will take three times as much.

Note: Actually the total time will increase even more because of the time lost in context-switching and coordination between the multiple tasks. The higher the degree of *Multitasking*, the more time is lost in this way.

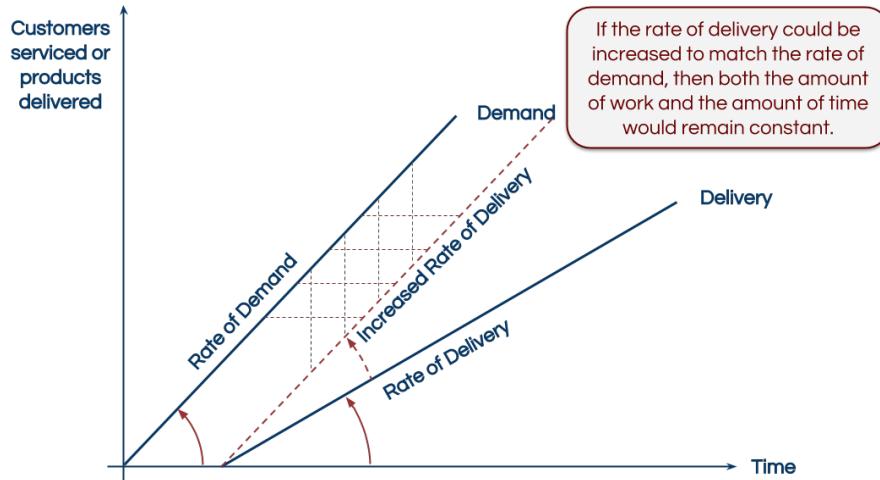
The Business Value of Matching Delivery to Demand

If we combine the learning illustrated in the recent diagram with the geometry of earlier diagrams we can infer the following:

- If the delivery rate matches the demand rate (i.e. the two lines are parallel), then the amount of work remains constant (rather than continually increasing).
- Likewise, the amount of time to finish any given work would also become constant (rather than being ever-increasing).

In other words, the system would become stable (constant amount of work) and predictable (constant amount of time). For a client - and all participants of an economic system - this is essential for maintaining trust and confidence, because the organization becomes more reliable and trustworthy.

It could be illustrated as follows, where all the horizontal and vertical segments between the demand line and the increased delivery line have the same (geometric) distance.



Yet, no matter how much companies invest in increasing their capacity, this ideal balanced and stable state seems forever unattainable.

Could there be another way to make those two lines parallel and thereby avoid overburdening the system, while at the same time making it both stable and predictable?

Thinking about the Demand Line instead of the Delivery Line

It is natural to think about acting on the delivery line and “*increase its slope*” because that is within our *Span of Control*. It is also natural to want to make the delivery line parallel to the demand line, as that would resolve many issues.



Span of Control: that part of our reality over which we have complete power to change anything.

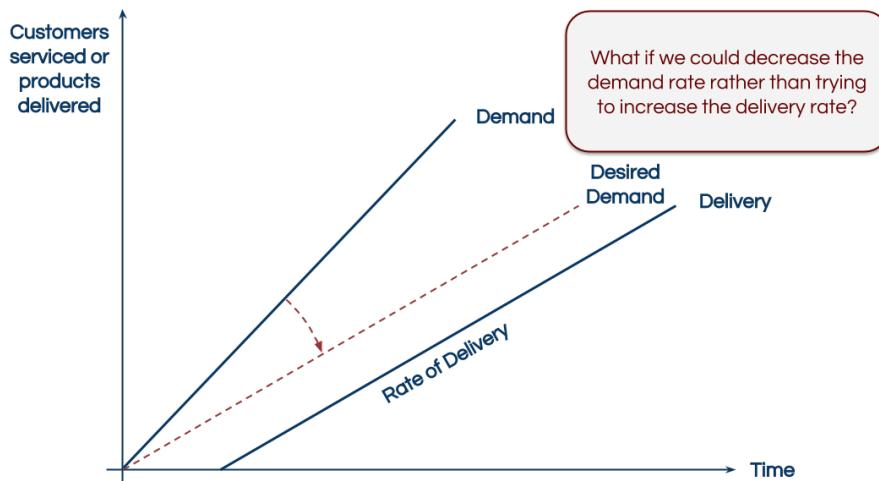
However, there are other possibilities, where the focus is not on trying to affect the delivery line. Specifically it is about “*demand shaping*” and **deliberately decreasing the slope of the demand line**, which is under our *Sphere of Influence*.



Sphere of Influence: that further part of our reality over which we have some kind of influence. There is no point working on something over which we do not have at least some influence - but in the *Sphere of Influence* we can resort to convincing arguments to win over the action of others, so that they affect us positively.

For instance, one could decide to stop serving the less profitable customers or market segments, and focus only on those providing high value. Thus demand would decrease.

On the diagram, it would look like this:



Of course, in most places, the idea of serving less customers or a smaller marketplace is not very popular. This idea has no merit - or does it!?

Moreover, in knowledge-work, the demand line is often (artificially) steeper than what it really needs to be. It becomes extremely hard to make the case for reducing the demand, when there are many stakeholders who all claim some critically vital reasons to be immediately served.

The Quest for Stability and Predictability

Whether we try to increase the rate of delivery or decrease the rate of demand, both approaches strive to achieve a condition where the two lines are parallel. When the lines are parallel, the system is not overburdened and, above all, becomes stable and predictable.

Being stable has huge business value. A stable system has greater odds of being around next year.

Likewise, being predictable is at the core of being able to “*keep our promises*” whenever we make them to our clients or other stakeholders. A predictable system is at the foundation of trust - and the premise of business relationships, since you do not do business with someone you do not trust.

But it seems that achieving this ideal condition is next to impossible.

On the one hand, increasing the rate of delivery is hard, it almost certainly incurs costs, takes time and is (mostly) prone to failure. (Note: the reason why it incurs costs and has high odds of failure will be further explored with the *Mental Models of Constraints Management*.)

On the other hand, decreasing the rate of demand is mostly out of our *Span of Control*, and is rarely accepted as a wise and viable business decision.

Yet, there is value in having those two parallel lines.

Is there another way to achieve this?

Unintended Consequences of Good Intentions

To find this other way, we need to question some of the assumptions and consequential beliefs, decisions and actions, that typically occur.

Given that the rate of demand is greater than the rate of delivery, the organization is literally under the pressure of the *Work Load* it is facing. Decreasing demand is out of the question; and demand pressure invariably results in more and more work entering the system.

One recurring refrain is: “*We have customers waiting to be served, we cannot let them wait! We must start working on their requests ASAP, so we can confidently reassure them they are being served!*”

The flawed *Mental Model* at work here is that in order to be responsive and to best serve customer needs, we must immediately jump in to action and start working as soon as there is an incoming request.

But it is this very decision that creates the instability in the system!

At the same time, there is this other flawed *Mental Model* that since most work is already late, the sooner you start serving a customer’s demand, the faster the work will be delivered.

These are the two assumptions that we need to reconsider:

- To best serve the customers, we must be able to tell them we are serving them immediately.
- The sooner you start, the sooner you finish.

Is it possible to think otherwise? Let’s find out in the next chapter.

Takeaways

Common sense informs us that organizational performance comes from better execution. However, prior to and during execution, decisions are made. Our subjective understanding and interpretation of reality will shape our decision making. Speed of decision making is paramount; but even more so is making the *right* decisions. Our understanding and interpretation of reality is shaped by our *Mental Models* - hence adopting more powerful *Mental Models* will allow us to improve performance in many ways: in making the right decision; making them quickly; and then executing them to the maximum effect.

One key tenet of the *TameFlow Approach* is that to achieve superior organizational performance, we must create the conditions of *Unity of Purpose* - otherwise our people will be pulling in different directions and waste energy and time simply fighting one another. The deeper insight here is that if we want to attain such *Unity of Purpose*, then all actors on our scene must share the *same Mental Models*. The prime directive for top management thus becomes: decide which *Mental Models* are relevant, and then make sure that all your people fully subscribe to them.

Within this philosophical frame, the question of which *Mental Models* are relevant becomes critical. We start off exploring this topic by reasoning around *Flow* and how quantifying and visualizing the

relationship between demand and delivery can affect critical decisions, that will bring considerable business benefits. The desire to increase performance, mitigate the presence and impact of *Multitasking*, and gain explicit awareness of current decision making drivers are the starting point for this exploration.

Chris Matts says...

Buy this book, you will get your investment back from the first chapter alone! The rest is a bonus.

Chris Matts, Agile Coach, Co-author of “*Commitment: A Novel about Managing Project Risk*”

Troy Magennis says...

Theory of Constraints can be boring. “Tame your Work Flow” by Steve and Daniel makes TOC practical and easy to understand - and (almost) fun.

Filled with actual techniques to observe and improve the economics of delivery, this book is essential to making the right decisions about flow and improvement.

Troy Magennis, President, Focused Objective LLC

2—Postpone Commitment and Limit Work in Process

As more and more work is pushed into the system, the amount of multitasking increases exponentially. Any work that is started is put aside in order to serve some other urgent request; then that work is resumed; then it is put aside again; then resumed; and so on and so forth.

The pattern is repeated over and over again. For all and any kind of work. High queue states beget high wait times; and high wait times beget high transaction costs which then beget even higher queue states all over again.

It is a vicious circle!

At 90% capacity utilization and according to Donald Reinertsen:

- We will find a resource readily available only 10% of the time
- Our average *Flow Time* will be 10 times the value added time
- 90% of our time in the process will be queue time !



Capacity utilization : The extent to which an enterprise uses its installed *Productive Capacity*.



Flow Time : *Flow Time* is also known as *Time in Process*, *Process Time* or *System Lead Time*. It is the time taken from the moment that work is started till it is finished.

Note: *Flow Time* is often and ambiguously referred to as *Cycle Time*, especially by practitioners of the *Kanban Method*.



Queue Time : The total amount of time it takes for a job to actually be attended to before value adding work is performed.

One way to address this issue is to focus on any one *Work Item*, and make sure it is worked on until “done” before another is started. The focusing mechanism can be achieved by deliberately striving to actively limit the *Work in Process*, purposefully eliminate *Multitasking* and adopting a sensible queueing or pulling policy.

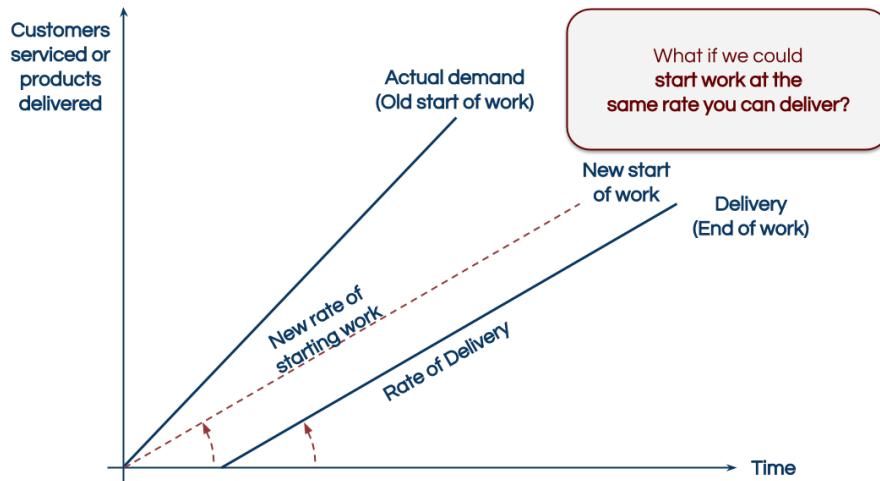
Limiting Work in Process

How can we deliberately *limit work in process*? Simply and literally by not starting any work until there is sufficient available capacity to handle it. In other words, it is just a matter of learning to **make the right decision about when to start work**.

Why would this make sense? First, it is a way to try *not* to change the rate of delivery (which typically requires investments, increased operational expenses and is also highly prone to failure).

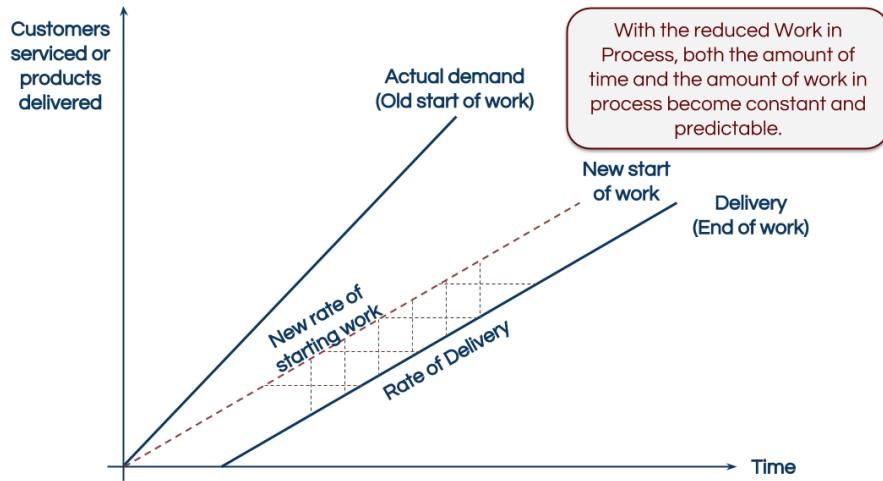
Second, by deliberately deciding when to start work, we can determine how much is loaded into the system. In particular, we can **decide to start work at the same rate that we are able to deliver it**.

In our simplified diagram, it would look like this:



The effect of not starting work is that the amount of work being worked on, the *Work in Process*, is reduced and limited; and because we now strive to start work at the same rate at which it can be delivered, we have the sought-after effect: *the two lines are parallel*.

Note: In the above diagram, don't make the mistake of believing that the distance between the *Actual demand* line and the *New start of work* line corresponds to what, in *Agile* parlance, is called the *Backlog*. As we will see in *Chapter 18 - Full-Kitting as an Ongoing Executive Activity*, the *Backlog* is a state that comes before what we consider as an *In-Process* activity.

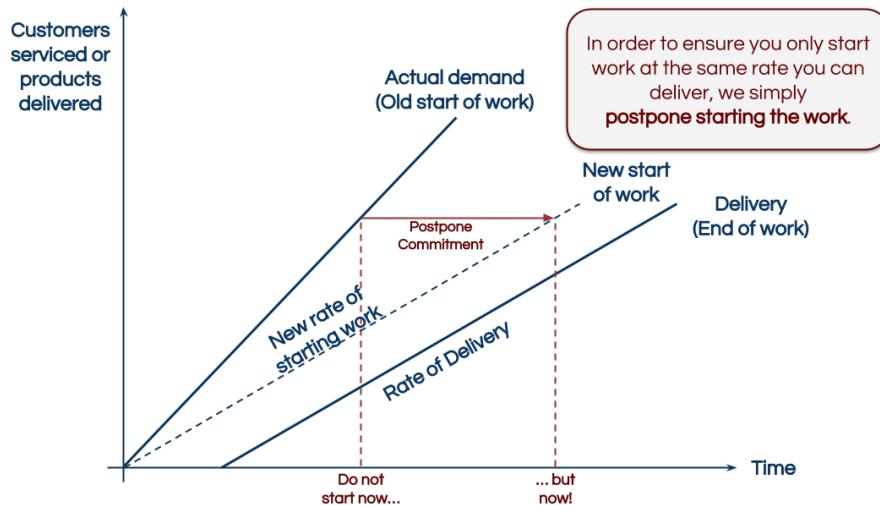


The amount of work in the system remains constant, as well as the time from start to finish. The system is not overburdened, but it has become both *stable* and *predictable*. We can keep our promises!

Postpone Commitment - But We Cannot Wait!

How can we achieve these two parallel lines, notwithstanding that we have ever increasing demand entering the system? One way is to **postpone commitment** (i.e. the start) of work.

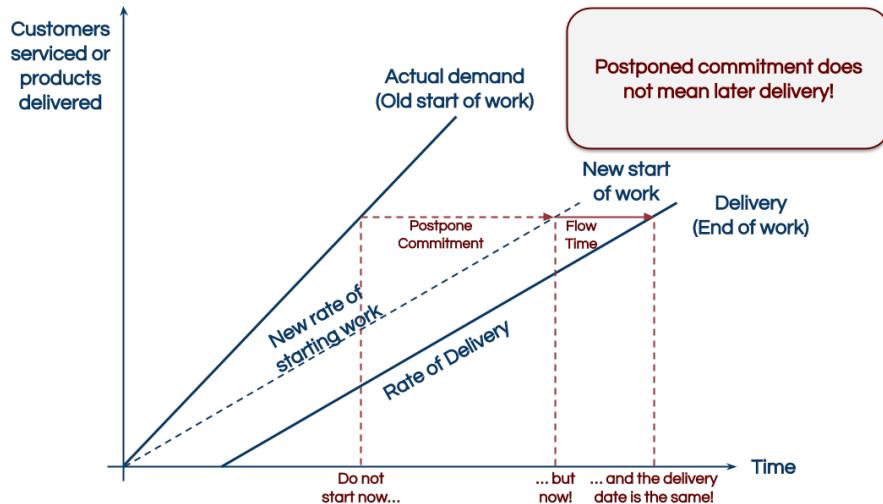
Visually it would look like this:



The very notion of *postponing the start of work* is at first hard to digest when you have unmet demand outside the system; and especially if accustomed to always starting work for the sake of “*serving the customer*” and being “*responsive*.¹ Our brains are unable to grasp that we need not increase capacity in order to fill unmet demand sooner. It is a handicap caused by a flawed *Mental Model*.

Even worse, if we believe that starting work sooner will deliver the work faster, we might conclude that this postponement will result in even later delivery, and hence discard the idea before thinking any further.

Yet the diagram below tells a different story:



Postponed commitment does not mean late delivery. Since we have not changed anything with respect to the slopes of the demand and delivery lines, the overall performance will – at least – not be worse off than before. (As we shall see, it will actually be better due to reduced multitasking and increased *Flow Efficiency* - which will be discussed in the following chapter.)

The effect of the postponement can be considered as a rearrangement of the time the *Work Item* is actually worked on, and the time the *Work Item* is sitting outside the system waiting to be handled. It is as if (most of) the waiting time is deliberately moved in front of the work.

But there is more to it than meets the eye in this simple transformation.

We are also reducing *WIP inside the system* with this stratagem. In fact - almost instantly - the *Wait Time* that was previously witnessed while performing the work inside the system goes down dramatically as *WIP* goes down. And we are reducing the level of *Multitasking* too.

The immediate impact of this decision is that we have now entered the realm of real options: work that has not been committed to, sits outside the system and becomes an option, which we might have the luxury to jettison without incurring any sunk cost loss. When discarding a *Work Item* that is already in process, we admit to having created waste because of our angst-ridden obsession with moving “fast.”

We really want work to be queued in a waiting line outside the system until it can be worked on and serviced with no interruptions once it enters the system.

The situation is similar to what happens when we stand in line for a burger at McDonald's or a ride at Disneyland.

It is a common experience - and a *Mental Model* - that we wait in line until our turn comes. We should ponder over why the fast food and entertainment industries understand this very well, while this is all forgotten in knowledge intensive businesses.

Selling the approach to the client in the light of “*waiting for your turn*” is totally in line with what we are accustomed to in real life, and it works very well.

From the delivery perspective, there is no apparent delay; there is no reason why a client would perceive this as any worse.

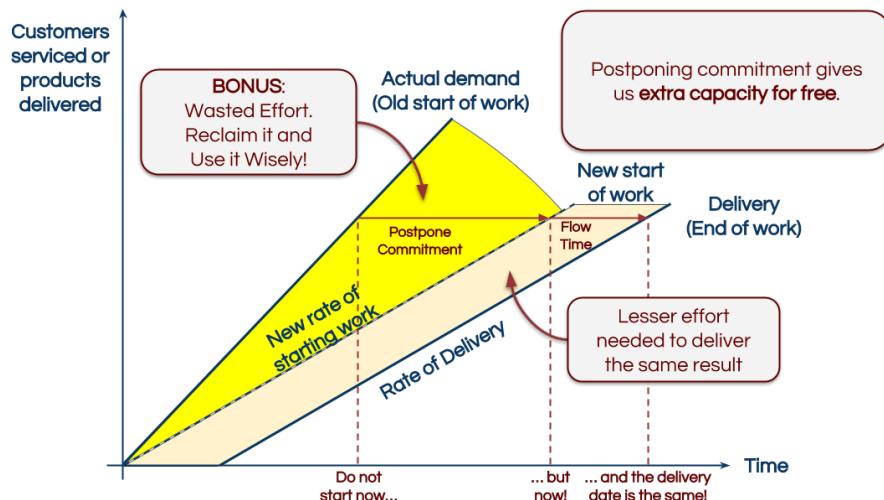
To make the process quantifiable and manageable, we can even set the clients' expectations and tell them:

1. how long they will have to wait before being serviced; and
2. predict when they will receive delivery.

This is very much like at McDonald's or Disneyland!

While there appears to be no material change from the client's perspective (because delivery still takes place at the same time), the positive effects are overwhelming: *we have gained the geometry benefits of those parallel lines that offers us stability and predictability. We will not be overburdened and shall keep our promises.*

But wait! There is an extra hidden bonus in doing this, as we can see by examining this figure:



The area that is highlighted in yellow corresponds to wasted effort. The effort is wasted because it is spent in order to *keep WIP* in the system (lost in *Multitasking*), rather than *getting WIP out* of the system. It is an effort that will just produce excessive *Work in Process* and *Multitasking* (unproductive “busyness”) without actually affecting how much is delivered - because the delivery line still has the same position and slope.

Note: The yellow shaded area does not represent the *Product Backlog*, which is still residing outside the system. The colored area simply consists of excessive *WIP* that causes *Multitasking* and is being removed in the new *Mental Model*.

We will revisit how this extra capacity bonus comes about in more detail in *Chapter 17 - Introduction to Full-Kitting*, where we will see one possible way how it can be put to good use.

Takeaways

Reducing *Work in Process* in a system must be appreciated at more than its face value as the following benefits are substantial:

- Reduces or even removes *Multitasking* from the system.
- Frees up capacity that ordinarily and primarily is wasted in *Multitasking*.

Just because we can free such capacity doesn't mean that we will deliver more (the delivery rate remained unchanged). In the next chapters we will see how to address this problem in a couple of ways. First by effectively reducing *Wait Time*; and second by elevating the *Constraint* in the process.

Diminishing *Work in Process* is also one of the recommended patterns discussed in *Chapter 21 - Patterns to Get Started!*

To conclude, the fact that the delivery line is under our *Span of Control* triggers the intuitive reaction to move it up to meet demand, which is only under our *Sphere of Influence*. This is a reaction that is engrained in the fabric of society and is assuredly one of the costlier flawed mental models in the corporate world today.

Kurt Häusler says...

Steve and Daniel have written a great book about knowledge-work management here, packed with practical information drawn from the Theory of Constraints that seems especially useful to Kanban users.

Release Train Engineers, Agile Project Managers and Product Developers will find ideas for new ways to use Kanban in areas such as portfolio management, at the value stream level and within product development, service delivery or devops teams.

This book could be a favorite of the Kanban, SAFe, and Scrum communities even if it doesn't suffer much patience for some of the questionable beliefs of these communities.

Like Reinertsen's Flow, this book is densely packed without much padding. Highlights include the chapters on Throughput Accounting and the actionable patterns in the back of the book.

Kurt Häusler, Kanban Coaching Professional, Enterprise Lean Kanban and Scaled Agile Consultant

3—Flow Efficiency, Little's Law and Economic Impact

We have learned that by deliberately deferring the start of work and matching the amount of *Work in Process* to fit our capacity, we have created the perfect zero cost pattern to get a stable and predictable system.

Now we can start to address the real issue of *increasing performance* - still at zero cost.

Touch Time and Wait Time

Before forging ahead, we need to adopt a common frame of reference for dealing with the many definitions of “time” in a flow perspective.

We can start by distinguishing the two states of *Touch Time* and *Wait Time*.



Touch Time: when work is being performed (“touched”).



Wait Time: when the work is waiting to be worked on while it is in process and no progress is made on it.

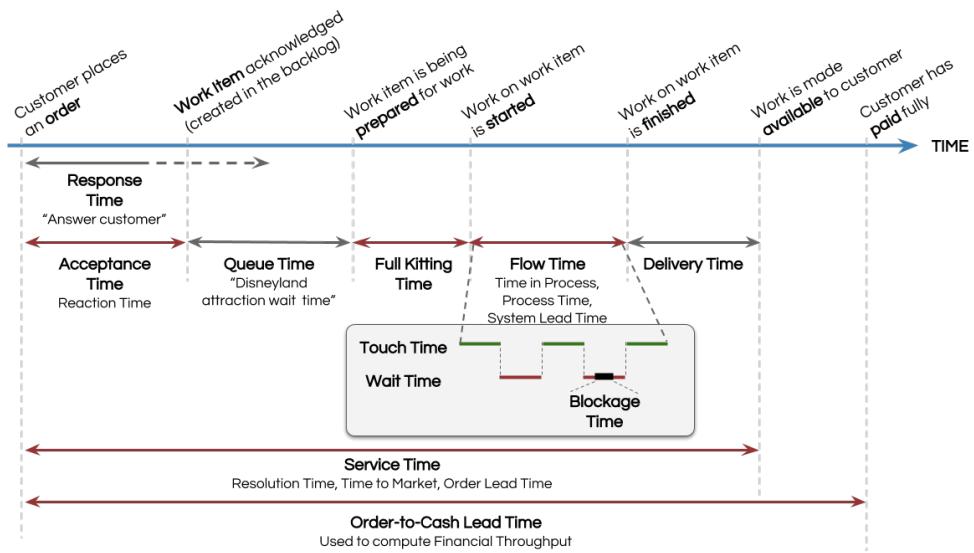
Touch Time and *Wait Time* are intrinsic parts of *Flow Time*. In fact all of *Flow Time* is partitioned in either *Touch Time* or *Wait Time*.

Some further points to keep in mind are:

- *Touch Time* cannot possibly exist outside our *Span of Control*.
- *Touch Time* and *Wait Time* - within the *Flow Time* boundary - are the metrics to be used in the computation of *Flow Efficiency*.

Note: *Flow Efficiency* and how it is calculated will be defined later in this chapter.

Understanding thoroughly the illustration below is important to sustain the high-level pace of discussion that follows in this book.



Even after extracting the “Disneyland attraction wait time” from the overall service time, there is still lots of time that goes wasted. We have weeded out all *Wait Time* that is *external* to our process, but there is typically substantial *Wait Time* *inside* the process (inside the *Flow Time*) itself.

Controlling the *Wait Time*

We need to understand what “eliminating” the *Wait Time* external to the *Work Process* truly means. (The context will be clearer at the end of the chapter, once we will have discovered and understood the value of the *Flow Efficiency* metric.)

There is a whole sector of this industry that prides itself on how it improves *Flow Efficiency* - but if we then examine how their “flow efficiency” is calculated - we note that it includes the *Wait Time* that is outside of the actual *Work Process*. This is a mistake. Using *Order Lead Time* or *Service Time* as the denominator for the calculation of the *Flow Efficiency* is not useful for operational improvements as neither are entirely within our *Span of Control*!

Note: Naturally *Service Time* is of major importance when considering the *Customer Experience*, but that is a topic for another book.

So what we are trying to do here is to set up the conditions whereby we can correctly measure *Flow Efficiency* in away that is useful to improving things within our *Span of Control*. In that sense we cannot “eliminate” the *Wait Time* that is outside of the *Work Process* but we need to understand that we must strive to eliminate the external *Wait Time* from the calculation of *Flow Efficiency*.

It does not mean that the external *Wait Time* implodes to nothing, nor that we are oblivious to its presence and/or effect on customer experience. It does not mean that we do not care about customer experience. Only that customer experience (“Disneyland” wait time) is simply out of scope if we want to deal with improving the *Work Process*, in particular via the *Flow Efficiency* metric.

Besides, even though the external *Wait Time* might be out of scope, if we are able to improve the performance of our *Work Process* many times over, then definitely the customer experience clearly improves as well.

Getting the *Flow Efficiency* Calculation Right

A great deal of literature on *Flow Efficiency* does not concern itself with the notions of *Span of Control* and *Sphere of Influence*.

Before going any deeper into the topic, it is necessary to consider how to correctly calculate *Flow Efficiency*. Unless there is a clear distinction of what is *in-the-process* and what is *out-of-the-process*, it is too easy to distort the metrics to the point that it becomes a vanity metric.

One very common error is to include what we define as *Queue Time* (or “Disneyland Attraction” time) in the calculation.

But it gets worse - because often even the *Delivery Time* is included in the calculation.

In particular, the proponents of the *Kanban Method* use their [*Order*] *Lead Time* (or *Time to Market*) to calculate *Flow Efficiency*.

The impact is that the wait component becomes disproportionately inflated, and the pre-improvement *Flow Efficiency* reported is much lower and much worse than it really is. Consequently, the reported improvements will be inflated and look much better than they actually are.

It is important to make the distinction between the *Wait Time* that is inside the actual process (i.e. inside *Flow Time*) and the “wait time” that lies outside of the process. The reason is that what happens inside the process is decidedly under our *Span of Control*, while what happens outside of the process is more likely only within our *Sphere of Influence*, and at times not even that. We have powers to act in our *Span of Control*, but not in our *Sphere of Influence*.

Many practitioners argue that “wait time” is always “wait time;” and that we may not have complete control over what happens during *Flow Time*. The point is not to argue that it may happen and that (due to *Special Cause Variation* only) exogenous events break the flow of the process and our being in control of it. The point is to realize that *outside* of the process we are definitely *not* in control in the same way because that ‘kind’ of “wait time” is **invisible** to the team executing the *Work Process*. However, *Wait Time* inside of *Flow Time* is different for it is very **visible**, obvious, tangible, and measurable; and it will get the managerial attention it deserves on a daily basis!

Wait Time and Options

Another critical weakness of this flawed way of calculating *Flow Efficiency* is that it is in contradiction to the precept of *leaving options open*. If *Work Items* are considered as real options, and the time to exercise the option is kept open for as long as possible outside the system, it really does not make sense to include that time in the *Flow Efficiency* calculation. Either, (1) the item has been committed to (the option has been exercised) and we are concerned about how efficiently we move the option from *Commitment* to delivery or (2), the item has not been committed to yet (the option is open) and we do not even know if work will be performed; and therefore we do not need to have efficiency concerns about speculative work that is *not* being performed.

Furthermore, there are motivational and visual elements in the *TameFlow Approach*. It is one thing to include in *Flow Efficiency* calculations a “wait time” that is invisible and out of our control in “Disneyland,”

and another story altogether when the actual *Wait Time* is made visible and actually measured while in *Flow Time*, and not just lumped together.

The motivation is different: there is a deliberate effort to make things visible, transparent and measurable. Consequently, the psychological drive to get things resolved becomes important. How this is actually done will be explained in *Chapter 14 - Flow Efficiency, DBR and TameFlow Kanban Boards*.

The popular perspective on *Flow Efficiency* includes “*Wait Time*” that cannot be seen, measured and managed; this is not comparable to how the *Tameflow Approach* actually calculates *Flow Efficiency*.

Note: Naturally there are different sources of “wait time,” which are industry specific. For instance in supply chain one would be concerned about: order, batch, queue, production, and transport, which *all look like “wait time” from the customer perspective*. And in large enough companies, there will be many layers of processes, which can play with various flow metrics in various ways. Here we are concerned exclusively with the *Wait Time* that is clearly inside our processes and falls within our *Span of Control*.

The Perspective of the Work Item

From the *perspective of the Work Item*, it is either being “touched” by a worker, or it is “waiting” to be touched by a worker.

Take patients in a hospital; either they are waiting in line for the doctor, or they are being “touched” (professionally, of course!) and examined or treated by the doctor (or some other qualified caregiver). A patient is called a “patient” for a reason; they must have a lot of “patience” waiting for their turn to be “touched” by the doctor.

Work Items passing through the *Flow Time* of a *Work Process* are like such patients. Either they are waiting for someone, or someone is working on (“*taking*”) them. They must also have a lot of patience, because most of the time, they will be... “*waiting*” for someone!

We want to change to a situation where *people are waiting for work*, rather than *work waiting for people*.

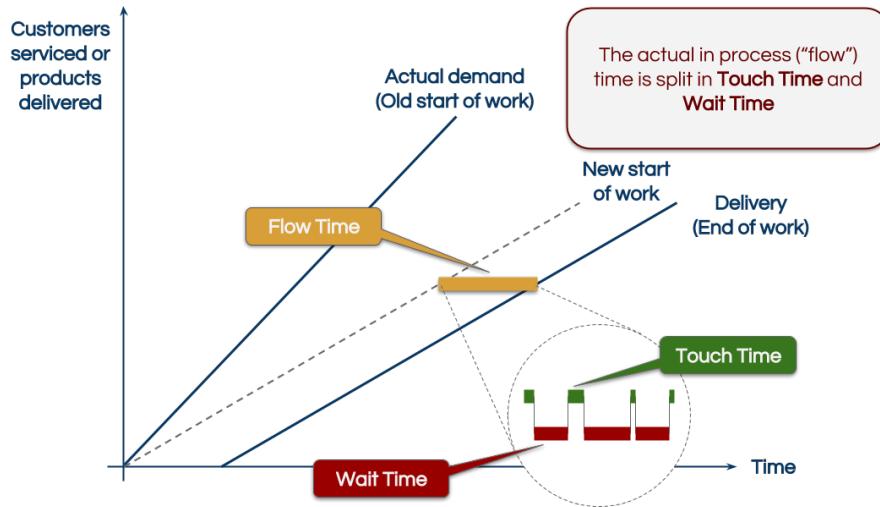
If people are waiting for work, we can have good *Flow* (or *Flow Efficiency* as we will shortly learn). If not, then everyone will be waiting for someone - and ultimately, the business will be waiting to get paid for the delayed delivery of its products or services.

Thinking about Touch Time and Wait Time in Practice

In our simplified model a horizontal line represents the time it takes for a piece of work to go from start to finish. This period of time is what we define as *Flow Time*.

Note: Remember what we stated earlier: in a *Cumulative Flow Diagram* the horizontal line will be the *approximate average time* it takes for an item to move through the process. In the rest of the chapter, always bear in mind that any horizontal line segment representing *Time* should be considered as an *average* or even an *approximate average*.

We can depict the *Touch Time* and *Wait Time* components of the *Flow Time* as follows:



The work timeline is split into two parts. In the illustration, the four green segments at the top represent the *Touch Time*, while the three red segments at the bottom represent the *Wait Time*. Figuratively, the *Touch Time* is shown in green and the *Wait Time* in red. This is to symbolize that *Touch Time* is positive as the work is actually being performed during the *Touch Time*. On the other hand, *Wait Time* is still time where no value is added: the work is standing still - like cars in front of a red light - hence the representation with a red segment.

Note: This also explains the choice of the term “*Work in Process*,” rather than “*Work in Progress*.” A piece of work can be *inside the process*, yet it can be standing still in *Wait Time* and, therefore, not making any *progress* at all.

If *Touch Time* and *Wait Time* are actually measured, it is not uncommon to find a disproportionate imbalance between the two, where the total amount of *Wait Time* is far greater than the total amount of *Touch Time*. It is no coincidence that the bottom segments are longer than the top segments in the illustration above. In knowledge-work settings, the disproportion would be even greater - so much so that the drawing would have to extend for several pages in width to represent the relative lengths of these segments!

Even without actually measuring *Touch Time* and *Wait Time*, we all have anecdotal evidence that this discrepancy exists. For example, we might hear of large organizations, like banks, that experience overall

time from request to delivery of software functions along the order of several months; while the actual work performed by the engineers takes only a day or two. This is due to the fact that work is mostly standing still waiting in queues for someone to actually do something!

Work Faster or Deliver Sooner?

Conventional improvement initiatives focus on “*working faster*” (or “harder”) because they aim at increasing the rate of delivery of work. In order to deliver faster, *Investments* have to be sustained, changes have to be undertaken, and the risk of failure is great. Improvement initiatives typically lead to colossal undertakings that often have little to show for in terms of hyper-performance compared with 300-500% returns that are possible with the *Tameflow Approach* and *TameFlow Kanban*.

Note: *Tameflow Kanban* is an alternative to the *Kanban Method*, since it evolved naturally out of the *Kanban Method*'s sixth core practice: “*Improve collaboratively, evolve experimentally, using models and the scientific method.*” It is based on merging: (1) The *Kanban Board* from the *Kanban Method* which evolved into the *TameFlow Throughput Board*. (2) The original Kanban system of the Toyota Production System, implemented with the Kanban tokens, used mainly to limit *Work In Process*, enable pull, and implement pull signals. (3) The *Drum-Buffer-Rope* (DBR) scheduling system typically used in production management to ensure optimal feeding of the constraint, to pace the release of *Work Load* into the *Work Flow*. (4) The use of *Buffer Management* to produce leading signals of unfavorable variability in the *Work Flow*, *Work Process* or *Work Execution*, and to ensure that the *Constraint* is exploited and that the system is subordinated to the *Constraint*, as suggested by the *Theory of Constraints*. (5) The introduction of the replenishment token, with the purpose of signaling pending capacity availability on the constraint implementing the “rope” part of *DBR Scheduling*. All of these elements will be presented and examined in later chapters.

We can observe that any complex *Work Process* is a sequence of interdependent steps, and that execution time is necessarily divided between *Touch Time* and *Wait Time*.

Working “*faster*” means being more efficient and “*touching*” the work less; focus is decidedly on reducing the *Touch Time*.

But we do have another option. We can consider the alternative.

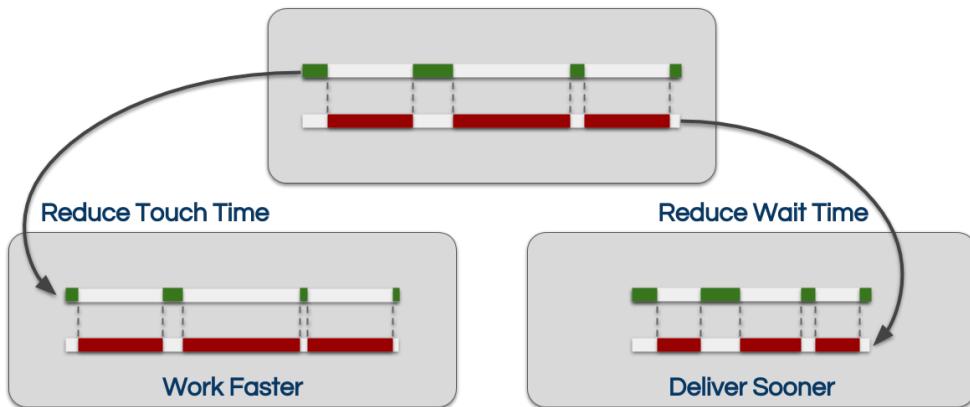
We can envision striving to reduce the *Wait Time* rather than trying to *work faster*. If we are able to *reduce the Wait Time without changing the existing Work Processes*, it means that we do not have to sustain *Investments*, undertake changes nor incur risks. One would be challenged to think of any other instance where smart money is no money!

The operation will be much cheaper – it will cost nothing! – and there will be no pain in adopting different *Work Processes* (because there are no different *Work Processes*), and no risk of failure because all work is still executed and performed exactly as before. It is also well aligned with the “*Start from where you are now*” Kanban principle!

Note: Kanban systems are economic systems aimed at matching demand to capacity. This is achieved

by observing the current situation, applying methods to improve and then measuring to ascertain that we are obtaining satisfying results. A Kanban system always uses “*Kanban Tokens*” (real or virtual) to show, pull and control the *Flow* of work in a capacity restrained production environment..

The two alternatives can be visualized like this:



It is worthwhile underscoring the effect of reducing *Wait Time*: everything remains as it was. We preserve the same *Work Processes*, the same tools, the same people, the same skills, the same infrastructure, etc. Hence there are no additional costs to sustain and virtually no risks of failure.

The one and only thing that changes is the decision about when to start and when to stop work.

In other words, the only thing that changes is how to *coordinate* and *synchronize* work with the deliberate purpose of reducing *Wait Time*.

But the actual *Work Process* is not changed at all. Simply effortlessly brilliant!

What is *Flow Efficiency*?

With the admonition stated earlier that we need to be wary about what we consider as *Wait Time* - and we contemplate only the *Wait Time* that occurs while work is *In Process* - when we start actually measuring *Touch Time* and *Wait Time*, we will typically find surprises.

The amount of *Touch Time* will merely be a small fraction of what we might initially expect it to be.

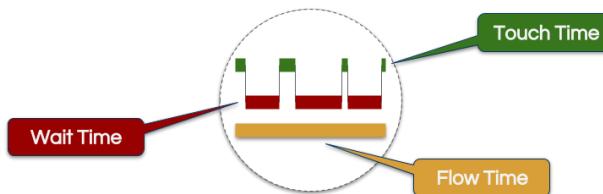
To contemplate just how much or little *Touch Time* there is with respect to the end-to-end *Flow Time*, we consider the metric of *Flow Efficiency*.



Flow Efficiency: the ratio between the total *Touch Time* and the end-to-end *Flow Time* expressed as a percentage.

$$\text{FLOW TIME} = \text{SUM(WAIT TIME)} + \text{SUM (TOUCH TIME)}$$

Typical **flow efficiency** is in the order of 3-7%. The easiest way to **increase flow efficiency** is to **reduce wait times**.



$$\text{FLOW EFFICIENCY} = \frac{\text{TOTAL TOUCH TIME}}{\text{FLOW TIME}} \times 100$$

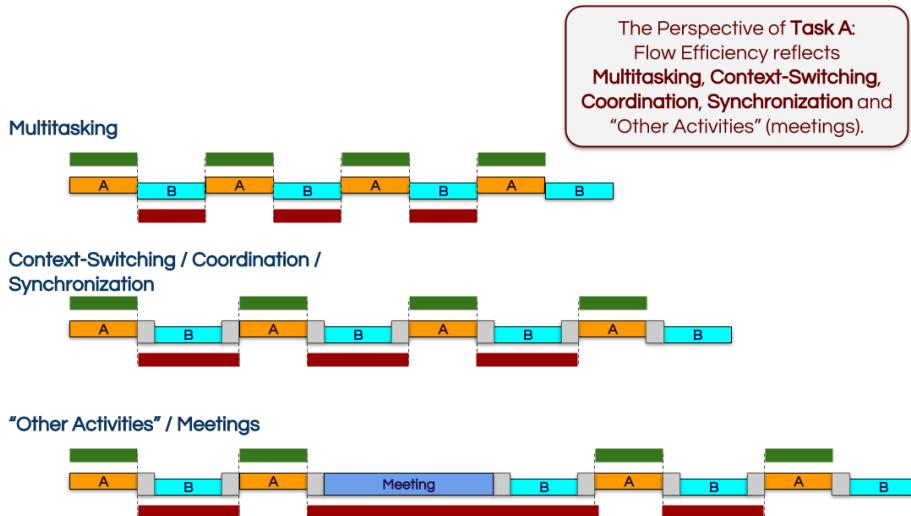
If no attention has been given to these ideas, *Flow Efficiency* is very low, especially in a knowledge-work setting.

Typically it is in the order of 3-7%.

Yes, you read correctly! 93 to 97% of the time your work is in a still state: idle time, feedback delay time, blocked time, wait time, induced work time!

Flow Efficiency, Multitasking and Other Time Snatchers

In Chapter 1 we already observed that *Multitasking* is undesirable. We can easily see how *Flow Efficiency* is a measure of *Multitasking* - and more generally of how much the *Flow* of work is disrupted by other activities. Here we revisit the diagram we saw in Chapter 1 to highlight how (from the perspective of *Task A*), *Flow Efficiency* gets gradually worse as more time is dedicated to other activities.



We show the *Touch Time* of *Task A* with a green bar above the process, and the *Wait Time* with a red bar below the process. Already, when *Multitasking* with only a second *Task* (and with this the simplified model where both *Tasks A* and *B* have the same duration), we can see that *Task A's Flow Efficiency* has dropped 50%: the *Task* is waiting for half of the time.

However, things get worse. Quickly. Switching from one *Task* to another typically incurs in context-switching, coordination and synchronization activities (shown with the gray bars in the illustration) which consume time - and will let *all the Tasks* at hand accumulate *Wait Time*.

Furthermore, people in companies have the habit of carrying out other activities also - such as meetings - which are one of the greatest time drains ever. While such other activities are ongoing, any *Work in Process* will incur *Wait Time*.

The red bars just keep on getting longer and longer.

Considering that the example shows only two concurrent *Tasks* while in most organizations there will be hundreds or thousands of concurrent tasks, it is easy to see how *Flow Efficiency* can degrade to single digit figures very quickly.

Hence the idea: if we strive to improve *Flow Efficiency*, we will gain more focus on actually getting work done. As we have already observed, in order to improve *Flow Efficiency* we have two ways: one is to try to reduce the *Touch Time* ("work faster"), and the other is to try to reduce *Wait Time* ("deliver sooner").

Since there is "more" *Wait Time* than *Touch Time* it sort of makes sense to aim at the former rather than the latter - and try to "deliver sooner" rather than "work faster."

But is it so? We need to investigate this further.

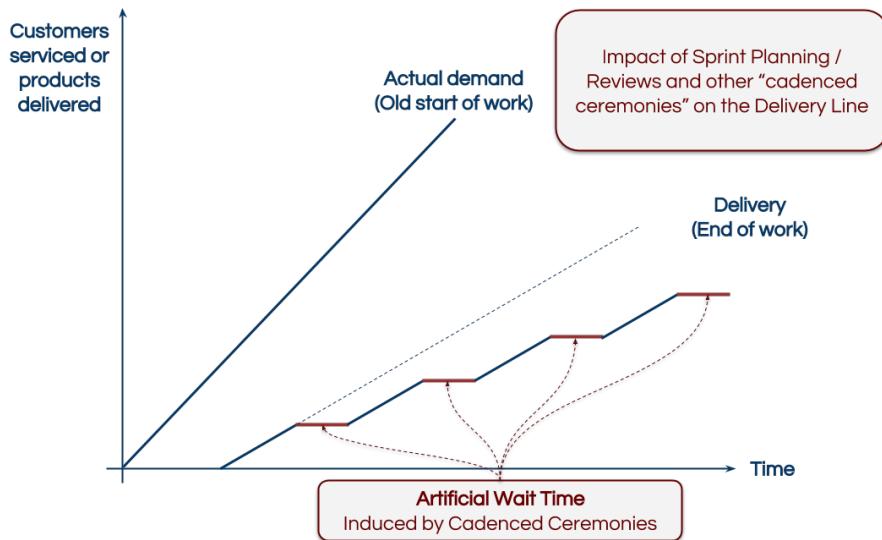
Beware of Sprints - A Big Time Snatcher

Before moving on, we need to reflect on the nature of *Wait Time* as it occurs in relation to *Sprints*, SAFe's *Product Increments* and *Iterations* as promoted by the *Agile* movement. All of these represent recurring and suboptimal operational habits that adversely affect *Flow Efficiency*.

In fact, *Sprints* artificially create contrived inflationary conditions between *Wait Time* and *Touch Time* states beyond what the normal course of operation would require.

This induces excessive context switching, coordination and synchronization costs that go against what Agile truly stands for at its roots: getting things done swiftly.

Sprints, and all timeboxing variations, have an inflationary effect on *Wait Times* as seen in the following figure:



When the typical Scrum meetings (*Sprint Planning*, *Sprint Review*, *Sprint Retrospective*, *Backlog Refinements* and even the daily *Stand-up*) take place, work is not being touched, and less time remains to actually “get stuff done.”

Ironically, Mullainathan and Shafir argue in their book “*Scarcity*” that creating artificial shortages of time make us lose a good portion of our IQ too! *Sprints* are in fact architected limitations that create such time deficits; and one has to ponder what value, if any at all, lies in the metrics gathered in a state of rush when fabricated time pressure manifests itself at the end of each and every *Sprint*!

What is *Little's Law*?

From the structure of the diagrams we have used in the first two chapters which plot *Work* against *Time*, we can also quantify the *Throughput* of our efforts in terms of the amount of work produced per unit of time. This is where we need to get familiar with *Little's Law* (named after Dr. John D. C. Little):



Little's Law: A theorem in queueing theory stating that the long-term average number of customers in a stationary system is equal to the long-term average effective arrival rate multiplied by the average time that a customer spends in the system. (Source: Wikipedia)

Or mathematically:

$$\text{LENGTH} = \text{ARRIVALS} * \text{WAIT}$$

We have to note two things here. First is that the relationship is between averages. We have to read: “the **average LENGTH** of a queue is equal to the **average ARRIVAL** rate multiplied by the **average WAIT** time.”

Second is that the law is expressed in relation to the *arrival rate* - not the *departure rate*. The *arrival rate* corresponds to the slope of the *Demand* line in work-time diagrams we have examined this far.

In our case, we will use an alternative expression of *Little's Law* which takes the form of this equation:

$$\text{TP} = \text{WIP} / \text{FT}$$

where *TP* is *Throughput*, *WIP* is *Work in Process* and *FT* is *Flow Time*. Notably, even in this second formulation, the three variables represent averages.

This second formulation is customarily used in operations management, in the *Kanban Method* and in the *Theory of Constraints* - and it is the one we will use throughout the rest of the book.

The major difference is that focus is shifted to the *departure rate*, or the slope of the *Delivery* line in our work-time diagrams. This formulation is easier to relate to, because business is more interested in “what gets done” (the *Delivery* line) in terms of *Throughput* rather than “what’s waiting to be processed” (the *Demand* line) in queuing theory - but it is really just a matter of perspective on the same underlying phenomena.

Note: The expression $\text{TP} = \text{WIP}/\text{FT}$ will give the impression that increasing *WIP* will lead to better results. This is true, but only if more work can be done for every unit of time; and therefore is the equivalent of “working harder” (and reducing *Touch Time*). In the following chapters we will see when and how this is a viable strategy; but in the meantime we want to focus on what it means to reduce *Flow Time* (by reducing *Wait Time* or *Touch Time* - or both).

Assumptions of Little's Law

When the spotlight is on the *Delivery* line, there are several assumptions that need to be fulfilled for *Little's Law* to be applicable. In particular, in relation to:

- the *Conservation of Flow*;
- the system's *Stability*;

- the use of consistent units.

Note: We will not elaborate further on the details of these assumptions, but simply refer to Dan Vacanti's "Actionable Agile Metrics" book for a comprehensive coverage.

Stability and Mental Models

No, we are not talking about mental health here!

The assumption of *Stability* can be thought of as the situation when - on our work-time diagrams - we can observe that the *Demand* line is parallel to the *Delivery* line.

This parallelism of the *Demand* and *Delivery* lines is a very important requirement not only for the applicability of *Little's Law*, but it is also an essential element in the elaboration of our *Mental Models*. In particular because of the following:

- In the next chapter we will reason a lot about the three variables of *Little's Law* and we will try to see how the presence of a *Constraint* will impact *Throughput*. We will elaborate *Mental Models* based on drawing the mathematical relationships between the variables of *Little's Law* as a triangle: *Throughput* will be represented by the (lower left) angle in such triangles. Such an angle will correspond to the *slope* of either the *Demand* line or the *Delivery* line - but only if those two lines are parallel. In that instance our geometric reasoning will be valid for either formulation of *Little's Law*.
- Later we will attempt to identify the *Constraint* in the *Work Process*. With the assumption of *Stability* in place, we will also observe long term stable average *Flow Times*, and in particular the *In-State Flow Times* will be stable. Stable *In-State Flow Times* will allow us to identify the *Constraint* in the *Work Process* by looking for the state with the longest average *In-State Flow Time* - but more about this in upcoming chapters.

Little's Law and the Theory of Constraints

The quest of this book is about finding the *Constraint* in a knowledge-work setting. While both *Little's Law* and the *Theory of Constraints* have been applied to similar domains - like manufacturing and operations management - there is little prior work that directly connects the two. We thus must be aware of the differences, which we can summarize - by oversimplifying - like this:

- *Little's Law* is stochastic and probabilistic and models the randomness of both arrivals and departures. It looks at long term averages. The main area of application is to reduce *Flow Times*.
- *Theory of Constraints* is simple and deterministic. It is very much about counting widgets delivered over a period of time, and finding the limiting factor - the *Constraint* - at the operational level. The main area of application is to increase *Throughput*.

As we just mentioned in the section above, in the next chapter we will resort to representing the relationships between the variables of *Little's Law* (that is: *Throughput*, *Work in Process* and *Flow Time*) geometrically, as a triangle and a slope.

We need to take the more deterministic view because we want to arrive at a *Mental Model* that caters for the presence of a *Constraint* while we still reason in terms of these three variables. The geometric reasoning we will expose must *not* be considered as an application or a reflection of *Little's Law*, notwithstanding that it relates to the same three variables, because in the geometric reasoning we do not consider the variables as long term stochastic averages, but as deterministic quantities.

Note: It is one of the author's [Steve] conjecture that the effect of a *Constraint* can also be modeled in stochastic terms - and thus the aforementioned *Constraint*'s based geometric reasoning would be made entirely consistent with *Little's Law* - though no mathematical proof of this has yet been found (to the best of the author's knowledge).

Little's Law and Flow Efficiency

We now enter a territory that is contentious. It might appear that whether we choose to reduce *Touch Time* or *Wait Time*, the time gained will be reflected in the denominator of the ratio that expresses *Little's Law* - because, in virtue of how we have decided to calculate *Flow Efficiency*, the *Flow Time* that appears as the denominator in *Little's Law* is the same that appears as the denominator in the calculation of *Flow Efficiency*.

Since the denominator is the sum of all *Touch Time* and all *Wait Time*, it stands to reason that reducing one or the other will result in a decreased *Flow Time*, and hence an increase in *Throughput*. Consequently, the relevance of the questions about where we should focus our effort. On *Touch Time*? On *Wait Time*? Or maybe on both? And in which order?

This territory is contentious and counterintuitive. As we will see in the following chapter, reducing *Wait Time* will not increase *Throughput* directly at all, even if it seems completely counterintuitive (though it does so indirectly in a very significant way).

We might conclude that focusing on *Touch Time* is the right thing to do - albeit, for the time being, we are not equipped to do so effectively.

Fortunately, we will also discover that focusing on *Wait Time* first has significant benefits; and we should consider it as a stepping stone towards eventually acting on *Touch Time* - but more about these details in the next chapter.

Economic Impact

The *Operational Throughput* of the system that is quantified through *Little's Law* can also be related directly to the *Financial Throughput* - or how much money the organization makes per unit of time.

We can examine the changes from a financial perspective by referring to the fundamental equation of *Throughput Accounting*.



Throughput Accounting: "A management accounting method that is based on the belief that because every system has a constraint which limits global performance, the most effective way to evaluate the impact that any proposed action will have on the system as a whole is to look at the expected changes in the global measures of throughput, investment and operating expense." (Source: TOICO Dictionary, Sullivan, 2012) For an introduction to *Throughput Accounting* see the blog post: [Theory of Constraints and Software Engineering](#).

The simple - yet powerful - *Throughput Accounting* equation is:

$$\text{ROI} = (\text{TH} - \text{OE}) / \text{I}$$

Here, the *Return on Investment* (ROI) is the difference between the *Financial Throughput* (TH) and *Operating Expenses* (OE), divided by any *Investment* (I).

We can reasonably expect the *Financial Throughput* to increase in proportion to the increase in *Operational Throughput*.



Financial Throughput: The money generated by the system minus *Totally Variable Expenses*.

It is here that we can appreciate that *Operating Expenses* would increase substantially when trying to "work faster" (i.e. striving to decrease *Touch Time*) for instance, if new people were hired. But there would be zero difference in *Operating Expenses* when trying to "deliver sooner" (i.e. striving to decrease *Wait Time*) because hiring or similar contingencies would never be required.



Operating Expenses: All the cash outlays necessary to transform *Investments* into *Financial Throughput*.

Investments would increase substantially when trying to "work faster;" for instance, new equipment would have to be acquired (computers, offices, etc.). But there would be zero difference in *Investments* when trying to "deliver sooner" because acquiring new assets or equipment would not be necessary - again because everything is still done the previous way.



Investment: All the money in the system. Excluding conversion costs which are part of *Operating Expenses*.

With this we are starting to see how *Flow Efficiency*, *Little's Law* and the fundamental equation of *Throughput Accounting* can tie together our time savings effort on *Wait Time* and *Touch Time* to the business's financial results.

Takeaways

All models are wrong, some are useful.

When two models from unrelated fields of science converge as *Little's Law* (queuing theory) and *Throughput Accounting* (scientific method and inherent simplicity) do, it is time to take stock as both models will support similar outcomes.

But is it really the case?

Let's remember that the *Kanban Method* has let go of *Throughput Accounting* and eventually rediscovered *Little's Law*. In contrast, *Tameflow Kanban* embraces both models.

To make the point, we need to go back to the first TameFlow book - the "Hyper" book - and examine the quote from Kurt Häusler appearing below:

"Kanban's background in product development and maintenance might focus on Lead Time where being quick to market is important, but many companies just need more things to be done in less time (rather than faster) and cheaper, which is where improving Throughput comes in."

Tameflow Kanban is profit centric and considers *Throughput* at the top of the *Flow* pyramid.

Understanding and mastering *Flow Efficiency* is lacking in the Agile world. *Touch Time* and *Wait Time* are not part of the Agile working vocabulary. This situation is rectified by *TameFlow* - as it is a cornerstone to attaining stellar sales, reducing costs and producing healthier *Throughout*. Being able to understand and connect *Little's Law*, *Flow Efficiency* and *Financial Throughput* will lead to superior performance. In the next chapter we will discover how to remove the obstacles that stand in our way in order to make that important connection.

Agile approaches like Scrum and its variants such as SAFe, Nexus, Scrum@Scale, also aim directly at impacting *Touch Time* and changing the way you work. They also inhibit a state of *Flow* by artificially propelling a high level of context switching between *Touch Time* and *Wait Time* due to the "timeboxes-and-ceremonies" design of Sprints.

Sprint, timeboxed or PI based approaches provide no direct proof of profit enhancements by any quantitative measurement aside from anecdotal evidence. Anecdotes are not a sound basis for sharp business decisions.

Marcus Hammarberg says...

This book is packed with experience, tested practices and ways of applying the principles for not only Agile, Lean and Kanban but maybe foremost, the Theory of Constraints. It is very nice to see this applied in a real-world setting and see how much great arguments and solid advice that can be deduced from the (relatively) simple principles and practices. Many good tools and visualizations are being shown and explained throughout the book and I can absolutely see how these can be used at my clients right away. I particularly like the ideas about optimizing/improving wait time rather than the working time. Much easier! And cheaper. And a better effect.

The book is based on the authors' experiences and opinions and they are on display throughout the book. This is not good or bad but something that should be noted before reading. Some readers like that others do not.

All in all, I learned a lot of new tools and gained a deeper knowledge of how to apply these ideas in practice from this book.

Marcus Hammarberg, Author of “Kanban in Action” and “Salvation: The Bungsu Story: How Lean and Kanban saved a small hospital in Indonesia. Twice. And can help you reshape work in your company.”

4—Utility of Flawed Mental Models

In the previous chapter, after becoming familiar with the concepts of *Wait Time* and *Touch Time*, we outlined how *Flow Efficiency*, *Little's Law* and economic results are all related. However, the question about whether it is more convenient to concentrate improvement efforts on *Touch Time* ("work faster"), on *Wait Time* ("deliver sooner") or on both, remained unanswered.

Once we will have learned about the intricacies of the *Theory of Constraints*, we will discover beyond any doubt that there is a lot to gain by reducing the *Touch Time* - and more specifically doing so on the *Constraint* - but that it is very difficult to do so properly.



Theory of Constraints: "A holistic management philosophy developed by Dr. Goldratt that is based on the principle that complex systems exhibit inherent simplicity, i.e., even a very complex system made up of thousands of people and pieces of equipment can have at any given time only a very small number of variables – perhaps only one (known as a constraint) – that actually limit the system's ability to generate more goal units." (Source: TOCICO Dictionary, Sullivan, 2012.)

However, as we progress toward achieving that insight, we can still gain a lot by first focusing on reducing *Wait Time* alone. Presently, given the nature of knowledge work, we are not able to effectively act on *Touch Time*. The reason is that in order to properly affect *Touch Time*, we must first perform *Step 1 - Identify the Constraint* of the *Five Focusing Steps*. For the moment, we are not equipped for that task.

Identifying the *Constraint* seems like an impossible task to accomplish in a modern knowledge-work setting. It was one of the reasons why proponents of the *Kanban Method* decided to forgo all of the TOC. Instead they opted to focus on alternative and less than optimal means to manage both the *Work in Process* and the method to detect where to intervene - specifically by introducing *Column WIP Limits*. (As we will see in later chapters, there are many reasons to refute this approach.)

However, we are still unable to identify the *Constraint*. We need to get there, somehow. This is where some expediency will come to our help and will subtly create the conditions that we need.

To get there we will make use of a *flawed Mental Model*.

Flawed Mental Models are not Necessarily Bad

In many pages of this book we criticize mainstream *Mental Models* that are inadequate; we propose alternatives, which we believe are a better fit for our purpose. However, we will now examine a *Mental Model* that we know is flawed - but we also know it will allow us to get closer to our objective.

We need to be very open-minded when using *Mental Models*. The adoption of a model has to be gauged against how well it allows us to make better decisions. And once a newer, better model is uncovered, then we can switch to the new one without dogmatic attachment to the old one.

A well known example of this is how Newton's *Law of Universal Gravitation* was replaced by Einstein's *General Theory of Relativity*. While it can be claimed that Newton's *Law* was wrong, it has nonetheless been immensely useful; and probably is still more useful than Einstein's *Theory* for many practical purposes and applications. We can agree with George E. P. Box who said: "Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful."

So let's examine this model that is wrong, but useful to guide us towards better decisions; and in the process prepare the ground for allowing us to identify the *Constraint* in knowledge-work settings.

A Good Wrong Reason to Accept a Flawed Mental Model

The game plan: we have observed that reducing *Wait Time* will not incur any additional costs or risks, while reducing *Touch Time* definitely does. Therefore, if we can show that reducing *Wait Time* provides us with tangible benefits, then we will have the perfect recipe to get something for nothing. It seems too good to be true.

The reasoning we will uncover will lead us to believe that such benefits are indeed possible - therefore, we should strive to reduce *Wait Time*. Paradoxically, the reason we will give to justify doing so, will turn out to be wrong - the flaw in the model - notwithstanding that it will guide us towards the right direction. We will discover that the benefits are even greater than what the flawed *Mental Model* will lead us to first believe.

This is where the utility aspect comes into the picture: if we actually pursue the reduction of *Wait Time*, we will *indeed* enjoy such benefits predicted by the wrong theory. In reality, as we will discover, there are secondary effects that produce even greater benefits. Those secondary effects are not caused or explained by our original, flawed reasoning.

One might wonder, why not start immediately with the right perspective to begin with? The answer is simply that it would not seem accessible or credible enough to get things started.

And we need to get started this way, because unless we first focus on reducing *Wait Time*, we will never be in the operating conditions to focus on *Touch Time*, which is where the real big gains are to be found.

Misunderstanding the Effect of *Wait Time* on Little's Law

Since *Flow Efficiency* is an operational metric, it must at least fall within our *Span of Control*. We want to answer the question of whether it is worth improving *Flow Efficiency* - and more specifically whether it is more worthwhile to focus on reducing *Touch Time* or *Wait Time*.

To reply factually, we would need to run some experiments and collect real data and then make the right decisions. This is typically what is done in *TameFlow*: we form a hypothesis, run experiments, collect data and decide accordingly - basically we fully adhere to the *Scientific Method*.

Short of being able to run a real experiment, we can run a mathematical *thought experiment* or a simulation; and then reason about a hypothetical improvement.

Suppose that we are given the power to effectively produce a time reduction in the order of 20%. But we have to choose where to apply it. We can choose to decrease the *Touch Time* (i.e. we increase the working rate, this is the conventional improvement thinking of “work faster”); or to decrease the *Wait Time* (i.e. postpone commitment, and “deliver sooner”).

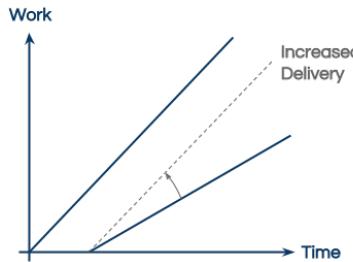
Suppose further, for the sake of argument, that the current *Flow Efficiency* is 5%. That means that 5% of the whole *Flow Time* is *Touch Time*; while 95% is *Wait Time*.

It is obvious that *any* time reduction will be reflected in the *Flow Time* that appears as the denominator in both the expression of *Little's Law* as well as in the calculation of *Flow Efficiency*.

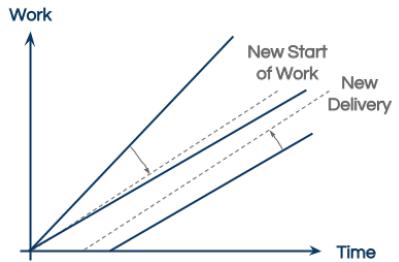
Naturally, since the simulation intends to show the difference in impact between the two cases, i.e. of reducing the *Touch Time* vs. reducing the *Wait Time*, we need to perform two calculations, reflecting the two scenarios.

Here is a breakdown of the reasoning in the two hypothetical cases, keeping in mind the hypothesis that we have a *Flow Efficiency* of 5%:

Work Faster



Deliver Sooner



| | | |
|---|---|---|
| 20% Reduction on Percent of Flow Time Flow Time reduction Flow Time becomes Throughput increase factor Impact on Throughput Collateral conditions | Touch Time Touch Time is 5% 20% of 5% = 1% $100\%-1\% = \text{99\%}$ of the original $100/99 = 1.01$ +1% Requires Investment, increased Operating Expenses and increased risk | Wait Time Wait Time is 95% 20% of 95% = 19% $100\%-19\% = \text{81\%}$ of the original $100/81 = 1.23$ +23% Requires no Investment, no change in Operating Expenses, and no new risks |
|---|---|---|

Alleged Economic Benefits for the Wrong Reasons

It appears that a 20% reduction applied on *Touch Time* is a much worse alternative than an equivalent 20% reduction applied on *Wait Time*, due to the (generally) bad *Flow Efficiency*.

Note: As we mentioned, low *Flow Efficiency* is typical of knowledge-work. In other domains, depending on the situation, *Flow Efficiency* might be medium to high. All of our reasoning is focused on the knowledge-work scenario.

The apparent conclusion is clear: *when reducing Touch Time by 20%, Operational Throughput will increase by a mere 1%; but when reducing Wait Time by the same amount, Operational Throughput will increase by 23%*!

The difference is astonishing and counterintuitive. But the situation can be said to be even worse because in the former case we have to sustain expenses, reorganizations, restructuring, retraining and must accept all related risks of failure; while in the latter, we incur none of those. We could justify this by resorting to *Throughput Accounting*. We could consider that the “working faster” option will need to be

sustained by an increase in *Operating Expenses* and/or by additional *Investments* (or, more likely, both), despite the nominal improvement of 20% in the *Touch Time*, the overall *Flow Time* improvement becomes a mere 1% reduction, while the bottom line impact will be almost negligible because the *Financial Throughput* will also take into account the negative impact on *Operational Expenses* and *Investments*.

It seems that there is no doubt that it is much wiser to try the counterintuitive approach and “deliver sooner.” Time to delivery will be reduced by 19% and *Operational Throughput* will increase by 23%; there will be no changes in how the work is performed; and consequently no increase in either *Operating Expenses* or *Investments*.

The positive outcome of this approach seems to be unbelievable: zero change, zero cost, zero investment, zero risk; but 23% more in *Throughput* and 19% better delivery time. And, evidently zero “resistance to change” as well, because there will be no survival anxiety (will they keep me?) or learning anxiety (will I be able to learn the new stuff?).

Basically, the improvement seems to come for “free” – except, of course, that we need to adopt the new *Mental Models* (determining when we start and stop work; resisting the temptation to decide when and where we invest to improve; and reconsidering how we view idle time or excess capacity) and make decisions accordingly.



CAUTION: The positive outcomes are real - even better than what they appear here - but unfortunately all of the above reasoning to explain how they come about is not correct!

Let's now discover why the model is flawed, notwithstanding that it does indeed move us in the right direction.

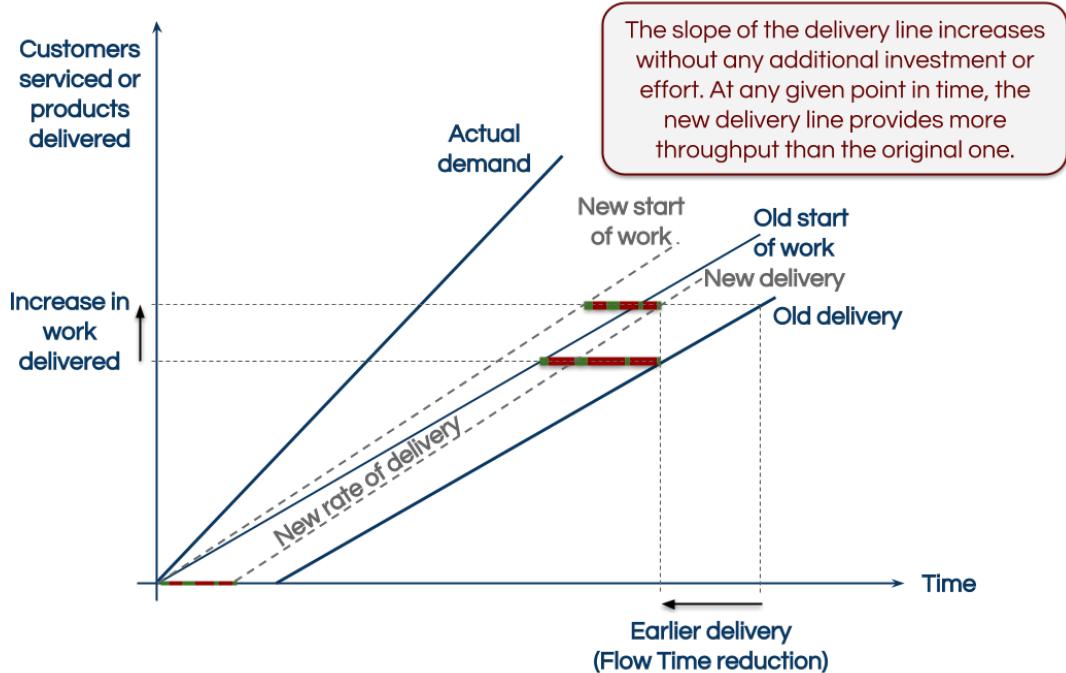
While we have been diligent to reason in terms of *Little's Law*, *Flow Efficiency* and the principles of *Throughput Accounting* in a very consistent, logical and mathematical way, we have failed to take into account what the presence of the *Constraint* does to all of this.

We will address this shortcoming in a moment. But first, let us reflect further on the utility aspect of what we have done.

Why the Flawed Model has Utility

We have found a great (wrong) reason to strive to “deliver sooner” instead of “working faster.” What becomes really puzzling, is that if we actually try to “deliver sooner” rather than “working faster” we will effectively witness a very significant increase in *Operational Throughput*.

If we measure and plot our findings in terms of our work-time diagrams we will conclude that reducing *Wait Time* will successfully improve *Flow Efficiency* and *Throughput*; and we might see something similar to what is shown in the following illustration:



While we might observe what is depicted above, the explanation - that the increase in *Throughput* is due to the reduction of *Wait Time* is wrong! The observed increase in *Throughput* is *not* due to the logic we have exposed above. While it is real and tangible, it is caused by much more subtle interactions.

As presented so far, the model will lead us to believe that the reduction in *Wait Time* results in the delivery line increasing its slope with respect to the original one. At any given point in time, the new delivery line will be “steeper,” “higher” and “more to the left” of the initial one.

We will believe that the slope of the new delivery line has increased, all the while work is not performed any faster or differently than before (because we deliberately “deliver sooner” rather than “work faster”).

The amount of work delivered at any point in time will be greater than before the *Wait Time* reduction effort. And, because we still keep in place the policy of starting work only at the pace at which it can be delivered, even the start of work line will see its slope increase correspondingly. All this while no effort or investment are expended to try to work any faster, work is still performed at the same speed and in the same manner as before. It seems too good to be true.

Therefore, we could stop here and be happy to reap the benefits of putting into action what we believe the model is explaining. The positive results would reinforce our belief that the model is indeed very sound - just like all practical experiments initially cemented the belief in the correctness of Newton's *Law*.

But we want to do even better - now we want to see Einstein's *Theory* to better understand what is really happening.

The Shortcomings of the Model

There is a fascinating paradox with which we must come to grips. Because of the deliberate choice to act on *Wait Time* alone, actual work is not performed any faster than before (and there is no additional stress or burden put on the system or on the people), and naturally the time from start to finish is still reduced. In fact, this is the very reason why this alternative is so attractive: the avoidance of costs and risks.

According to *Little's Law*, the *Flow Time* at the denominator decreases, and thus *Throughput* will increase. The contradiction here is that - from a *Theory of Constraints* perspective - a system simply cannot deliver any more *Throughput* than its *Constraint*. Since we have not reduced *Touch Time*, the *Constraint* cannot have increased its *Capacity* in any way.

Remember: the team of boy scouts could not proceed any faster than Herbie! So if we are not actually reducing *Touch Time* (e.g. making Herbie go faster), then how is it possible that there is an increase in performance? It should remain the same!

We must contend with the obvious contradiction that a system cannot deliver any more *Throughput* than the *Constraint*. The discrepancy here arises from the fact that while we can see an increase in *Throughput* both graphically (as we saw in the above diagram) as well as through *Little's Law* function, the very fact that we have acted only on *Wait Time* and not on *Touch Time* means that we have not changed anything in the *Work Process*.

In theory, the *Capacity* of the *Constraint* of the system would remain unaffected; and it would *still* be working at the same rate as before. Therefore, according to TOC, we cannot possibly see any difference in the system's *Throughput* - yet that is what we observe!

This inconsistency deserves some deeper thought and explanation.

There are two aspects we need to understand:

1. Reducing *Wait Time* does not directly affect *Throughput* in any way.
2. Reducing *Wait Time* will indirectly affect *Touch Time* and hence will indirectly increase *Throughput*.

Let's try to understand these two aspects.

Reducing *Wait Time* does not Increase *Throughput* (Directly)

When we are reducing *Wait Time*, we must realize that some *Work in Process* that previously was waiting must necessarily no longer be waiting at all. In other words, that some amount of *WIP* has exited the system - or rather we have prevented it from entering the system in the first place. This means that we are keeping more work waiting in Disneyland rather than being stuck inside the process.

The reduction of *Wait Time* will cause a proportionate reduction in *WIP*, and hence the *Throughput* as defined by *Little's Law* will remain unchanged.

Still, we must explain how it is possible that *Throughput* does indeed increase when we effectively reduce *Wait Time*. It is all explained thanks to the indirect effect on the *Constraint* and how we allow the *Constraint* to approach or even reach the maximum of its *Capacity*.

The Real Beneficial “Side Effect” of Improving on *Wait Time*

The effect of low *Flow Efficiency*, as found in knowledge-work, holds the key for understanding the resolution of this contradiction.

When *Flow Efficiency* is low, the system ordinarily performs at a rate that is *less* than that of the nominal *Capacity* of the *Constraint*.

Why?

Because a lot of time is spent in dealing with *Multitasking* with its high cost in terms of context-switching, coordination and synchronization.

When knowledge-workers are overburdened with more and more work, they have no choice but to handle it by switching from one piece of work to another, providing the illusion that the best performance arises from simultaneously working on all of those pieces of work together, so as to not “waste” time. In reality a lot of time is literally wasted in the context-switching, coordination and synchronization activities that happen when knowledge-workers resort to *Multitasking*.

Using the story of the Scouts to illustrate the situation of the *Multitasking*, Herbie would have to stop every few steps to do something unrelated to actually moving forward, like changing shoes (i.e. switching to other tasks). The scout leader would get very annoyed if that happened - but this is what ordinary operational practices not only accept but actually encourage and even institute in knowledge-work.

Focusing on reducing *Wait Time* is thus a way to remove *Multitasking* and to *exploit* the *Constraint* even without knowing where it is. It is not a coincidence that the second step of the *Five Focusing Steps* (5FS) is about “exploiting” the constraint.

Note: Notwithstanding the negative connotation of the word - to “exploit” - we are reminded here that the aim is to have the *Constraint* work at its maximum sustainable capacity, without disturbances.

This is a very important and subtle point. We are in fact achieving Step 2 of the *Five Focusing Steps* (5FS), i.e. we “exploit” the *Constraint* but *without knowing where it is!*

Remember, Step 1 of the *Five Focusing Steps* is to *identify* the *Constraint* - but in ordinary knowledge-work settings, we simply do not know how to do this. In fact, the quest of this book is to learn how to do so! Yet, here we are already acting on Step 2 without having done Step 1.

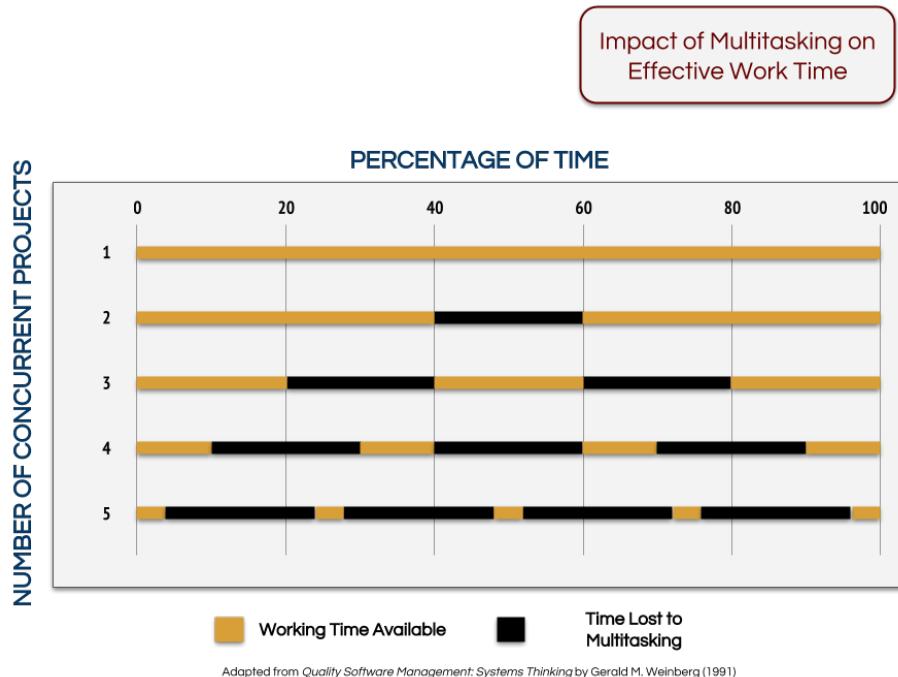
How is this possible?

Anything we do that allows the *Constraint* to work in optimal conditions equates to “exploiting” it - even though the *Constraint* itself might not have been explicitly identified yet (and even if we might not be fully aware of all the techniques of TOC). In other words, this is a way to act on the *Constraint* even if - for the time being - we do not have the means to identify it. It is a powerful insight full of operational and economic consequences.

Striving to reduce *Wait Time* is a simple means to reduce the amount of *Multitasking* across the system - and that applies to the *Constraint* as well. When *Multitasking* is reduced for the *Constraint*, the *Constraint* will waste less of its *Capacity*. Again: this is a simple way to “exploit” the *Constraint* without even knowing where it is.

The increase in *Capacity* is not really a proper “increase.” It is simply avoiding wasting the *Capacity* of the *Constraint* in unproductive *Multitasking*. And because we are not really increasing the *Capacity* of the *Constraint* but just avoiding its waste, we do not incur the costs and risks that are typical when we sustain an attempt to improve the *Touch Time*.

To better visualize how this is possible, let’s consider a well-known fact regarding the impact of *Multitasking* on the effective work time, as described by Gerald M. Weinberg in 1991 and illustrated in the figure below:



Here we can see that when focus is squarely on a single *Task* (or project as it may be), then 100% of a worker's available time is effective *Touch Time* for that single *Task*: it is *all* dedicated to work.

When we have five *Tasks*, then only 20% of a worker's available time is *Touch Time* - and it has to be shared among those five projects.

Therefore, if in the pursuit of reducing *Wait Time* we effectively go from 5 concurrent projects to 1, it is as if the *Capacity* of the *Constraint* "increased" five-fold (going from 20% to 100%). And because that *Capacity* is no longer shared by 5 projects but dedicated to one alone, we will have another five-fold increase. The effect is as if we "increased" the *Capacity* of the *Constraint* by 25 times!

Note: The above diagram shows the perspective of the doctor, not the perspective of the patient. In the case when there are five *Work Items* (patients!), the 80% of "work" time that is wasted in *Multitasking* by the worker (doctor) will count as *Wait Time* for five times: all the patients will be waiting for this overburdened doctor - that's why *Multitasking* is such a formidable multiplier of waste!

Again, for the sake of avoiding misunderstandings: the *Capacity* of the *Constraint* is not really "increased." Rather it is as if 80% of its *Capacity* that was wasted in *Multitasking* is recovered. That's why this way of proceeding is effectively *exploiting* the *Constraint* (in the TOC sense) even without us knowing where it is located.

So all of this ensures that the *Constraint* is working closer to the maximum of its *Capacity*, without having to change anything in the way work is performed, and without incurring the related costs or risks of change. We even avoid the case where the *Constraint* itself is *waiting for work* that is stuck and delayed or standing still further upstream, because of *Multitasking* happening there - and we know that one of the lessons of TOC is that the *Constraint* should never stand idle.

The increase of *Throughput* we observe is merely a consequence of “more work” produced by the *Constraint* simply because it is experiencing fewer interruptions (coordination and synchronization costs), not waiting for others to do their part, and reducing its own *Multitasking* (context-switching costs). The *Constraint* operates in a more focused manner, while the work is not performed “faster” or “differently” - only time is used more wisely.

The Importance of Proper Metrics

Another simple reason why the *Constraint* (and actually all steps of the *Work Process*) underperforms, is found in the fact that *Wait Time* and *Touch Time* are often not properly measured - so it is not known how bad the performance really is. Generally speaking, it is very difficult to measure the two variables accurately; but in most organizations there are little or no efforts undertaken to collect accurate metrics; even less to attempt to estimate or gauge the variables. Therefore, while everybody appears to be super busy, we have *Wait Time* slipping away as it is posing and masquerading as *Touch Time*.

When workers are *Multitasking*, what appears to be effective work time is in reality senseless cost and time spent on context-switching, coordination and synchronization. Time that should be accounted for as *Wait Time* but is not, because it is hidden away behind busyness.

If we are concerned about the efficiency of the *Constraint*, we must realize that it might not be working at the best of its *Capacity* due to insufficient instrumentation and measurement that prevents us from knowing about it. In *Chapter 12 - Flow Efficiency, DBR and TameFlow Kanban Boards* we will see how to properly capture and account for *Wait Time*.

Effects and Limits of “Deliver Sooner”

If we strive to “deliver sooner” we will achieve many benefits, which can be summarized by saying that *Work in Process* and *Multitasking* will be reduced. It will become easier and easier to keep the demand and the delivery lines parallel, and the system stable. With a stable system we will be able to be more predictable and keep our promises, and the conditions of applicability of *Little’s Law* will apply.

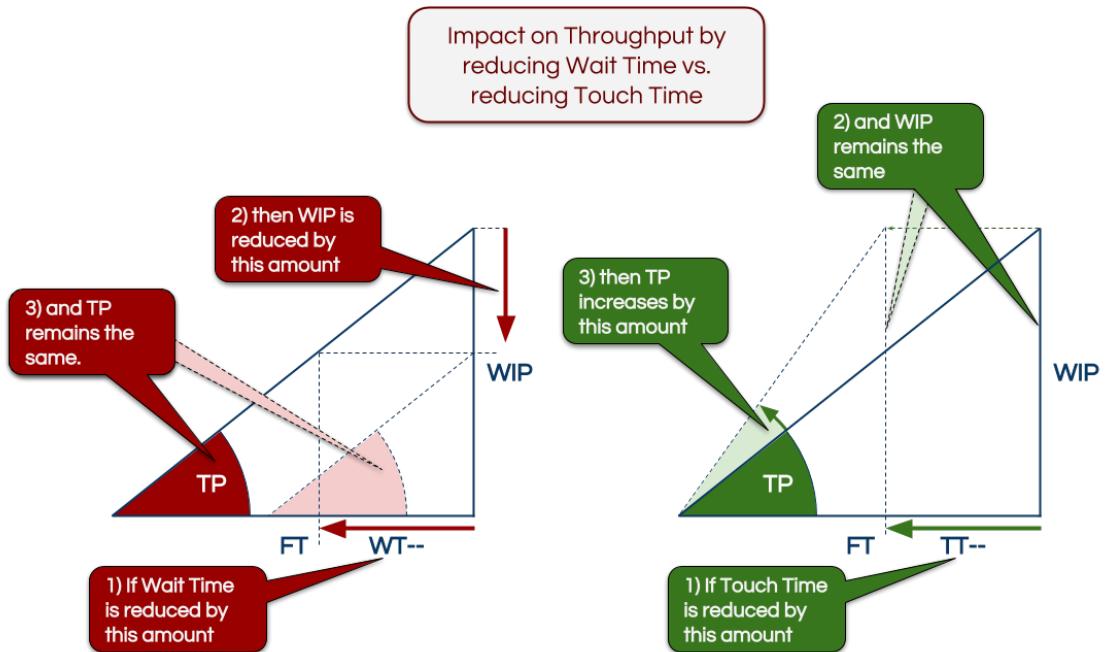
Note: *Little’s Law* should not simply be something that we learn “en passant” in our Kanban journey. It is so important to take it into consideration that without it, a lot of the basic benefits of our Kanban systems will never materialize.

Even more important: the conditions for reasoning around the *Constraint* are realized. Specifically, with a stable system, we can start measuring *in-state process times* in a reliable manner. As we will see in later chapters, this will allow us to properly identify the *Constraint* in the *Work Process*. So all of this is a necessary stepping stone in order to apply the rest of the thinking presented throughout this book.

Reducing Wait Time vs Touch Time

We need to clearly understand what happens when we reduce *Flow Time* by acting on *Wait Time* or on *Touch Time*. The two cases are illustrated in the figure below.

Note: The illustrations on these pages - with the “triangular reasoning” between the WIP, FT and TP variables - could be considered as a visualization of *Little’s Law* and its components. It is likely that - provided the assumptions of applicability of the *Little’s Law* are upheld - the following reasoning would still be valid even when considering all line segments and angles as averages of stochastic processes. However, for the sake of avoiding mis-representation and mis-application of *Little’s Law*, hereafter we will consider them as deterministic quantities and not as stochastic averages. This way we are more consistent with the *Theory of Constraints*’ way of reasoning about the presence and the effect of a *Constraint* - and in any case our reasoning can be validated simply by collecting proper *Flow* metrics.



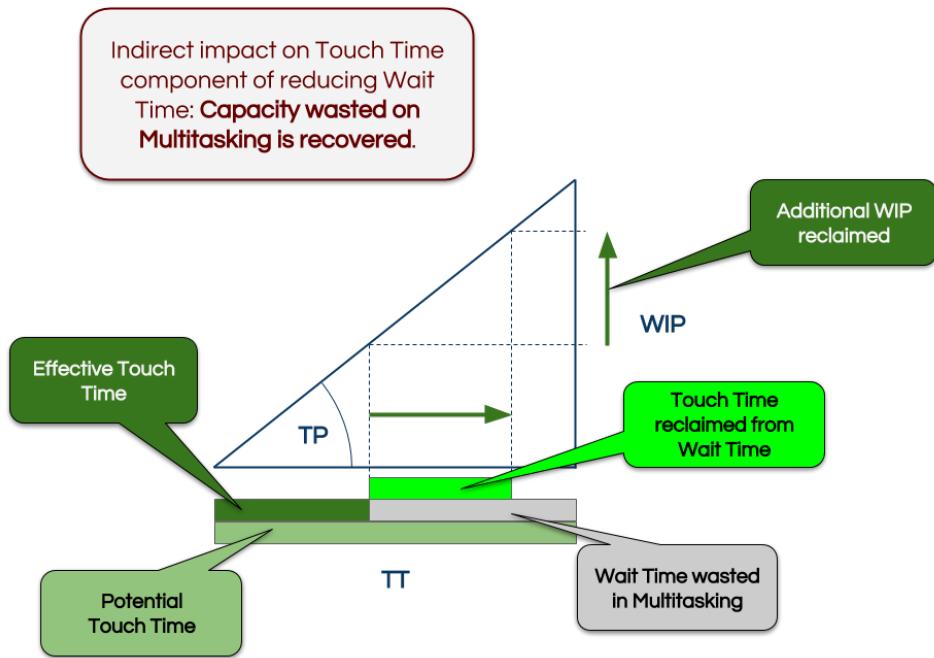
It is clear that we are visualizing the relationship between *Throughput*, *WIP* and *Flow Time*. We know that *Flow Time* is composed of *Wait Time* and *Touch Time*.

This is what happens:

- When *Wait Time* is reduced (“deliver sooner”), a corresponding amount of *WIP* is left out of the system, and *Throughput* remains the same. We experience a decrease in *Flow Time*.
- When *Touch Time* is reduced (“work faster”), then the same amount of *WIP* is processed in less time, hence *Throughput* increases. We experience, again, a decrease in *Flow Time*.

Effect of Removing Multitasking

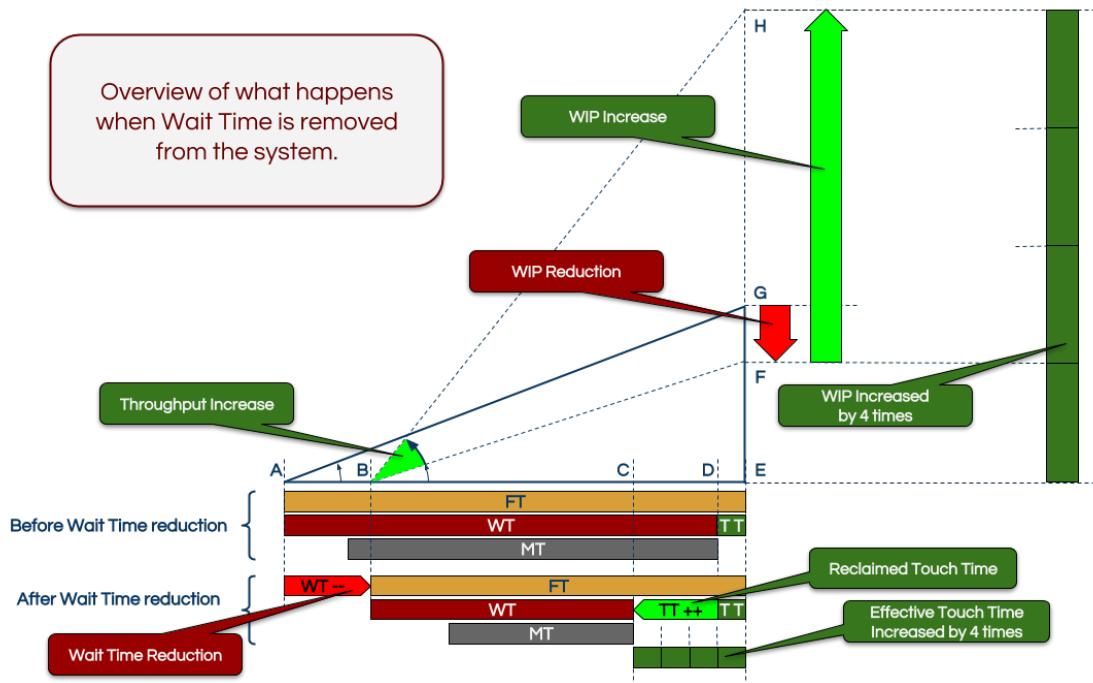
Once we remove *Multitasking* by reducing *Wait Time*, the impact on the *Touch Time* component of *Flow Time* can be understood as follows:



As we can see, the *Throughput* (the slope) does not increase, but because there is more time available as productive *Touch Time*, more work can be delivered. The increase in *Touch Time* is directly related to the time we reclaim from *Wait Time* that is caused by excessive *Multitasking*.

Overall Effect of Reducing Wait Time

Now we have all the pieces of the puzzle to understand what the actual effect of reducing *Wait Time* is. Let's consider the illustration below and bring together, in a geometric manner, all that we have discovered thus far.



We have named key points with letters of the alphabet, so we can refer to them and the segments and angles between them.

Our system is delivering a certain amount of *WIP*, represented by the segment [EG], over a period of *Flow Time* [AE]. The *Throughput* is the ratio between the two, represented by the angle $\angle EAG$.

Below the figure we represent the different times with bars. We see that *Flow Time* (FT) is made up of *Wait Time* (WT) and *Touch Time* (TT), respectively counting for 95% and 5% of *Flow Time*. In turn, the majority of the *Wait Time* component contains the *Multitasking Time* (MT). As we learned earlier, the sum of the *Multitasking Time* and the *Touch Time* represent the nominal maximum (time) *Capacity* of the system (or more specifically of the *Constraint*). At present the system is underperforming, precisely because of *Multitasking* - only a fraction of the maximum *Capacity* is effectively used to, well, "touch" the work.

Further below we see the same bars as they might appear after successfully reducing *Wait Time*. The actual *Wait Time* reduction (WT-) of 15% will lower *Multitasking Time* and reclaim more *Touch Time* (TT++) in proportion to 15%. We see that the *Wait Time* reduction is projected into the segment [AB], while the *Touch Time* increase corresponds to the segment [CD].

Let's see how the events unfold on the diagram. The *Wait Time* reduction [AB] of 15 % will also produce a proportional reduction of the amount of *WIP*, reduction that we see represented by the segment [FG].

Remember that when reducing *Wait Time*, *Throughput* remains the same: in fact we see that the angle $\langle EAG \rangle$ is the same as the angle $\langle EBF \rangle$. This is necessarily a consequence of the fact that the *Constraint* is not working any differently; hence *Throughput* must remain unchanged.

Since less WIP is in the system, *Multitasking Time* (MT) is reduced, and the *Wait Time* [AB] that is saved is proportionally reclaimed as further *Touch Time* [CD].

Note: The degree of this proportionality can vary, as was evident from the earlier diagram representing the findings of Gerald Weinberger. In the figure, for the sake of simplicity - remember, it is a *Mental Model* - we simply assume they are the same: the saved *Wait Time* [AB] is the same as the reclaimed *Touch Time* [CD].

Here we gain a key insight. Although it is yet another approximation of our *Mental Model*, it stands to reason that if the original *Touch Time* [DE] delivered a certain amount of *WIP* [EF] - after the *Wait Time* [AB] reduction - then the greater *Touch Time* [CE] - after the *Touch Time* reclamation [CD] - will deliver a proportionally increased amount of *WIP* [EH].

We observe that the new *Touch Time* [CE] is four times larger than the original *Touch Time* [DE]. Hence four times more work can be done - and we represent that by making the final *WIP* [EH] four times larger than the *WIP* [EF] we started with (after the *Wait Time* reduction [AB]).

This is how the *WIP* delivered by the system is “increased,” without actually “working faster,” all the while the *Constraint* is still working at the same pace as before - only that it will have *more time* to dedicate to active work (“touching”) during the same period rather than wasting that time in the circumspect juggling exercise that is *Multitasking*.

Note: To better understand how this happens without actually increasing the *Capacity* of the *Constraint*, let's consider again the story of the boy scout hike. Say that Herbie can walk at a maximum speed of 4 km/h. If he walks uninterrupted for one hour, at the end the distance covered is 4km. Now if we force him to stop every so often to change shoes - yes, it is absurd, but so is *Multitasking* - he will still walk at 4 km/h *in between* the shoe changes. If those changes happen very frequently (“multitasking”), after one hour Herbie might have covered only 3km instead of 4km. Yet his nominal *Capacity* (4km) was the same - and notably always fully used when effectively walking. If we instruct Herbie to cease with the changing of shoes - i.e. stop *Multitasking* - then the covered distance goes from 3km to 4km. But we have not done any *Elevation* - Herbie's maximum possible speed is still 4km/h; we have only eliminated the interruptions. (Later we can give him lighter equipment - that's elevation - and he might briskly walk 5km/h!)

In the figure, while inspired by what we saw in the earlier reference to Gerald Weinberger's findings, we are more conservative in the effects. We reduce *Wait Time* by [AB] to what amounts to 15% of the original *Flow Time*, and we make the simplifying assumption that the same amount of time [CD] is reclaimed as *Touch Time*. (If Weinberger's proportions were in effect, the amount of reclaimed *Flow Time* would be even greater; yet even this conservative assumption is sufficient for us to arrive at the concluding analysis.)

Remembering that the original *Touch Time* [DE] was 5% of *Flow Time*, by reclaiming the 15% saved off *Wait Time* [AB], we effectively have *four times* as much *Touch Time* as we had at the start: *Touch Time* goes from 5% [DE] to 20% [CE] for a 300% increase. Therefore, it stands to reason, that we can deliver four times as much *WIP* - from the baseline [EF] to the four-fold [EH] line - during the new reduced *Flow Time* [BE].

This multiplier affects the *WIP* [EF] that we have after the initial *WIP* reduction [FG], due to the reduced

Wait Time [AB]. So if we multiply the reduced *WIP* [EF] by four, we get the new *WIP* [EH] which we can see at the end. The effective increase in *WIP* is represented by the [FH] segment.

Note: One might object that this is not entirely consistent, because the original “unit of work” was the [EG] segment, while after it is the [EF] segment which is smaller. In either case we must not forget that the “unit of work” also accounts for *WIP* that is in the system, and that is actually *waiting*. Of course, if we removed all *Wait Time*, we would get to the actual unitary amount of work.

Naturally now even the *Throughput* is greater than the previous one, because the angle $\angle EBH$ is greater than either the $\angle EAG$ or $\angle EBF$ angles. And this explains how reducing *Wait Time* will produce an increase in *Throughput* - through a combination of the variables of *Little’s Law* and the impact of the *Constraint*.

Clearly, reducing *Wait Time* by [AB] has given us the greater *Throughput* of $\angle EBH$. We can see that even a relatively small reduction in *Wait Time* produces a significantly larger increase in actual total work [EH] delivered at the end.

This is why reducing *Wait Time* is so fruitful. We increase performance, without incurring costs or adding risks, because we are not actually changing how the work is performed during *Touch Time*, but we are amplifying the duration of *Touch Time* by reclaiming it from the *Multitasking* component of *Wait Time*.

To summarize: *Flow Efficiency* was initially 5%. We reduced *Wait Time* by 15%, and that saving was reclaimed as an additional 15% of *Touch Time*, which brought the new *Touch Time* to 20%.

If we normalize and suppose that the original *Flow Time* was 100 units of time (so we can more easily refer to these percentages), and keep on using such normalized units of time, then *Flow Time* (which was 100) becomes 85; the *Touch Time* (which was 5) becomes 20; and the *Wait Time* (which was 95) becomes 65. We can see these changes in the table below.

We see that *Flow Efficiency* goes from 5/100 to 20/85 - or, expressed in the customary percentage, *Flow Efficiency* goes from 5% to 23.53% for an overall appreciation of 371%!

Moreover, we now have 4 units of work delivered instead of 1. So *Throughput* goes from 1/100 to 4/85, which is an increase of 371%. As we can see, this is much better than what we had originally thought earlier in this chapter when considering *Little’s Law* alone and without knowing anything about the impact of the *Constraint* on *Wait Time* and *Touch Time* - where we concluded that a 20% reduction on *Flow Time* would deliver a 23% increase in *Throughput* in virtue of *Little’s Law* alone.

Summary of observed changes

| Measurement | Before | After | Differential |
|-----------------|--------|--------|--------------|
| Flow Time | 100 | 85 | -15% |
| Touch Time | 5 | 20 | 300% |
| Wait Time | 95 | 65 | -32% |
| Flow Efficiency | 5% | 23.53% | 371% |
| WIP | 1 | 4 | 300% |
| Throughput | 0.010 | 0.047 | 371% |

Clearly this almost four-fold increase in *Throughput* is explained only by taking into consideration the fundamental idea of the *Theory of Constraints* that a system cannot deliver more than its *Constraint*, and that any time that is lost at the *Constraint* is time lost for the whole system. This should give us the definitive motivation to try to reduce *Wait Time* before doing anything else - and tell Herbie to stop changing shoes!

Note: One subtle necessary condition to see this magnitude of improvement in *Throughput* is that the *Constraint* is truly inside the system and not in the market. In other words, there must exist sufficient demand to consume the greater amount of work that can be delivered by the improved system. We have not stated this explicitly as it was always implied by the greater slope of the demand line that we encountered in the initial chapters. We can safely say that this is an assumption that runs throughout this book. If we are in a situation where the *Constraint* is in the market (for instance if we are a startup launching a new product for which demand has to be created), then we need to resort to other approaches, and employ - for example - Dr. Goldratt's *Thinking Processes*.

When to Switch from *Wait Time* to *Touch Time*

The model presented offers many benefits. When making a deliberate effort to eradicate *Wait Time* and improve *Flow Efficiency*, there will come a point when there will be diminishing returns: the *Constraint* will have recovered all of its *Capacity* wasted in *Multitasking*. At that point the *Constraint* will not be able to deliver more than 100% of its *Capacity* unless we do something to “elevate” its *Capacity* (which typically corresponds to Step 4 of the *Five Focusing Steps*).

Even before we get to that point there will be no more *Wait Time* that can be effectively removed; so we will have no other choice than to reduce the *Touch Time* of the *Constraint* - that's when we have to actually consider the conventional option of “working faster” and focusing on *Touch Time* in order to truly increase the *Capacity* of the *Constraint*.

Note: In a system with interdependent processes and statistical fluctuations - as is the case of knowledge work - it is impossible to completely eliminate all *Wait Time*. In fact, as we will discover in later chapters, by using *DBR Scheduling* we will have a way to reduce *Wait Time* to the minimum that is necessary to prevent the *Constraint* from starving.

When we get to the point of deciding to focus on reducing *Touch Time* then it is also the moment we need to adopt additional *Mental Models* – those of *Constraints Management* – to learn exactly where to focus improvement efforts and where to invest in order to have a real impact on the organization’s performance, while minimizing the risk of failure. Beware of improvement approaches that do not focus on the *Constraint*.

The *Theory of Constraints* is very specific, as it teaches us that we should act primarily on the *Touch Time* of the constraint, rather than arbitrarily on all or any other step of the *Work Process*.

As we will see in the next chapter, reducing *Touch Time* can even produce no impact at all. We will learn that for any *Touch Time* reduction to be effective, it *must* happen on the *Constraint*.

There might be a few exceptions to this general rule. For example, we might want to increase the *Capacity* of the *Non-Constraint* before acting on the *Constraint* itself, in order to preemptively ensure that the *Constraint* will not move to some other place; and thus strive to keep the system stable (and keep the two lines, of start of work and of delivery, parallel).

Conversely, we might also do the opposite, just to gain stability. On the manufacturing floor, for instance, it is typical that before acting on the *Touch Time* of the *Constraint* for the first time, TOC practitioners will (as they say) “choke the release” or “freeze” in order to reduce *Work in Process* - thus effectively making the two lines, of start of work and of delivery, parallel as desired.

This is a key point in understanding the controversy. TOC does not have the problem of “reducing” *Wait Time* as we are experiencing here in knowledge-work, because TOC squarely “minimizes” and “controls” *Wait Time* before even starting - by “choking the release” in manufacturing environments or by “freezing” all projects in project environments.

Then *Work* is progressively and increasingly released into the system until the point that the full *Capacity* of the *Constraint* is reached; and thereafter work is released only at a pace that is equal to that *Capacity* - which is done through *Drum-Buffer-Rope Scheduling* (a technique we will employ too and that is described in later chapters).

TOC can operate like this because it acts on pre-existing flow systems - perhaps badly flowing systems, but yet flow systems where the *Constraint* can clearly be identified and known from the outset.

In knowledge-work we don't have the privilege of identifying the *Constraint* so easily. Thus we resort to reducing *Wait Time* first. The act will create the conditions of stability that allow us both to apply *Little's Law* and - even more significantly - be able to effectively measure stable *Flow Times* across the system and across the individual stages (through their *Wait* and *Touch Times*). Once we have achieved this, then and only then can we start to identify the *Constraint* and catch up with all other notions taught by Dr. Goldratt.

If we understand that we need to act on *Wait Time* first, we have to provide an economic justification. The *flawed* idea that reducing *Wait Time* increases *Throughput* in virtue of *Little's Law* can get us started in the right direction; and positive results will follow the actions. Yet we are better off by fully appreciating what happens in these instances: namely the side effect that is truly responsible for the *Throughput* increase that is the recovery of *Wait Time* wasted in *Multitasking* - wasted *Wait Time* that is reclaimed as productive *Touch Time*.

This approach works to a limit: when there is no more *Wait Time* to recover or - more likely - when the *Touch Time* is such that it employs the full *Capacity* of the *Constraint*, then we have to move on.

That's when things start to become difficult, because we will have no other choice than to actually improve *Touch Time* and truly increase the *Capacity* of the *Constraint*. Thus we need to know where the *Constraint* is.

Without knowing where to focus efforts, any undertaking will be totally in vain and wasted. Most improvement efforts centered on *Touch Time* will not produce any positive results at all. The only place where *Touch Time* improvements are beneficial is on the *Constraint* - but more on this in the next chapters.

Without proper *Mental Models*, such efforts are very likely doomed to fail or, in the most favorable conditions, produce only marginal improvements without any significant (financial) benefits.

Improving *Flow Efficiency* is a Low Hanging Fruit

We will see in the forthcoming chapters that to truly improve overall performance, we will have no choice but to act on the *Constraint*. Yet, what we have learned so far is that by focusing on reducing *Wait Time* we can already achieve very significant improvements, and basically without incurring costs or risks.

Naturally, there is a cost involved, but it is not a material one: it is the *psychological* difficulty of first being open and receptive to the new *Mental Models*, and second to humbly admit that the old ways of thinking were flawed.

This might be particularly relevant for executive management. Such executives might have had decades of successful careers, and might have a hard time seeing their own prevalent *Mental Models* as flawed. (For more on this particular aspect, see *Chapter 5 - Management's Responsibility and Learning Organization of the Hyper-Productive Knowledge Work Performance* book.)

One might wonder why everyone is not pursuing this since it is so wonderful. The answer is simple: the prevailing *Mental Models* do not allow to even *see* the alternative. It is a matter of awareness and mental dexterity. Once these new perspectives are revealed, they will be taken for granted and irrefutable. However, while they are *obvious* in hindsight, they are not (usually) *self-evident* to begin with. That's why, if there is a material cost involved, it might be in the form of training and riding a learning curve - which is still negligible compared to massive *Investments* and *Operational Expenses* that are otherwise involved.

Takeaways

In the last two chapters, we have challenged the manner in which *Flow Efficiency*, *Wait Time* and *Touch Time* are perceived - in particular by *Kanban* specialists - while Agile practitioners still have to truly

discover these notions.

But we do this with a purpose. Better decision making. How? By taking intuition out of the equation as we have shown the geometry and the mathematics behind *Flow Efficiency* and how *Wait Time* and *Touch Time* are “joined at the hip” when combining *Little’s Law* with the ideas of the *Theory of Constraints*.

The major misunderstanding that is addressed is the idea that improving *Wait Time* alone will increase *Throughput*. It does indeed - but indirectly - through the collateral effects that happen on the *Touch Time* side, by allowing the *Constraint* to reclaim effective *Touch Time* that ordinarily is wasted in keeping work waiting.

However, we must reckon that in knowledge-work we cannot jump straight into the application of all the concepts of the *Theory of Constraints*, because a necessary condition to do so, is to first identify the *Constraint*.

We need some preparation work to create a situation where that is possible. The tactical move is to act on *Wait Time* first (i.e. the way we decide to start and finish work); and aim at making the system stable and minimize the amount of *Work in Process*. Then we act on *Touch Time* next (i.e. the way, the “how” work is performed), but that has its own challenges as we will discover in the next chapters.

Multitasking – as we all know – is the poison that paralyzes the brain and the firm. Gerald M Weinberg called it the “context-switching tax.” *Multitasking* is wasteful both as transaction and coordination costs. Such costs are invisible. We cannot find a direct lineage between *Multitasking* and costs on the ledgers - but it is there. Yet we can find a direct link between the *Flow* metrics and the bottom line on the ledgers, as we will discover later on.

One of the greatest takeaways of this book is that we expose the causality between *Multitasking*, excessive *Work in Process* and *Flow Time* (and its components of *Wait Time* and *Touch Time*). It is always hard to argue the case for lower *Work in Process* as this is a major cause of resistance to change in knowledge-work. Tying the three variables in one solid model is essential in being able to make the case for a new “work order.”

If confronted again with a costly proposal for an enterprise wide improvement initiative that inevitably aims at reducing *Touch Time* first and changing the way we work, just stop and think: is that approach *Constraint conscious*? We will see in the next chapters the dire consequences of falling into the trap of “business as usual” when using the “usual approaches” to continuous improvement.

Daniel Couture says...

For several decades, remembering the Kaizen days in the Japanese industrialization, Organizations have focused on increasing productivity, and often times solely interpreted as cost reduction and optimizing bottom line.

This book provides a different look at workflow management and I take away two key messages that resonated with me: First, it is not only about optimizing throughput, it is as much about increasing revenue (demand) and matching work processes as it is about managing production flows. And secondly, Continuous Improvement methods and process improvements such as Lean, 6-sigma, Kanban etc, generally do not consider constraint management in their approach, and as a result will not generate the anticipated results, potentially wasting resources.

After having lived through countless - as well as fruitless – Continuous Improvement transformations as a CIO, I can only recommend to all C level executives to take a few hours to read this fantastic book and embrace the challenges that lie ahead with optimism.

Daniel Couture, CIO at UNICEF and former CIO at Zoetis and former CIO at Pfizer for several Global Business Units

PART 2—Acting on the Archimedean Lever that Boosts Performance: the Constraint

One of the most powerful *Mental Models* we can ever acquire is on how to manage the *Constraint* of a system. Yes, this is the *Story of Herbie*, revisited to gain a deeper understanding of the impact of a *Constraint* and how we should think anew.

We will discover that acting on anything but the *Constraint* is (with a few exceptions) mostly a waste of efforts and resources. We will learn that the resources that are *not* the *Constraint* will have to work at less than their nominal capacity; and that when we instead deliberately address and improve the performance of the *Constraint*, then the whole system's performance gets a boost.

The *Story of Herbie* bears many fruits. It is time to recognize them, so that they can be leveraged.

Al Shalloway says...

Perhaps the best way to describe “Tame your Work Flow” is that I now know I did not understand Dr Eliyahu Goldratt’s Theory of Constraints prior to reading it. Yes, I had read The Goal 20+ years ago. But even with reading follow up articles I missed the amazing power of the Theory of Constraints. The authors have undertaken two of the most significant challenges in improving organizations today. The first is gaining insights into how to deal with complexity and the second is teaching people how to use them. The TameFlow Approach does both.

When dealing with complex systems, people often attempt to simplify them. But even if that is accomplished, the systems are still enormously complex. The way out is to understand the Theory of Constraints’ principle of Inherent Simplicity. Inherent Simplicity is the presumption that inherent in complex systems there are rules that, when understood, enormously simplify the system. Inherent simplicity already exists. We must find it and take advantage of it. This will enable us to increase performance and reduce or eliminate the challenges we are facing.

This approach is used in the hard sciences. Consider the movement of all the bodies in the universe. Newton came, three rules. Now simple. Newton did not invent the rules. They were there. But he found them and now we know how to correctly predict the behavior present. It is true that some systems are so complex that even with an understanding of the inherent simplicity in them we may not be able to accurately predict their behavior. But with this understanding we can predict what will be improvements to the behavior of the system, and that’s what we’re going after.

Understanding the inherent simplicity in development systems is not enough, unfortunately. One must also convey this knowledge to people so that they can take advantage of it. In “Tame your Work Flow,” Steve and Daniel lay out many inherently simple concepts that they have discovered. Then they take these concepts and show us how to use them.

Al Shalloway, Director, Thought Leadership for Agile at Scale Programs, the Project Management Institute, co-author of “*Lean-Agile Software Development: Achieving Enterprise Agility*.”

5—Where to Focus Improvement Efforts

We have now seen that there are great benefits in trying to reduce *Wait time* rather than *Touch Time*. This will result in improved *Flow Efficiency*.

The main attraction of this approach is that in order to reduce *Wait Time*, we do not need to change anything except the criteria we use to decide when to start and when to stop work.

Compared to trying to reduce *Touch Time*, it is the most powerful way to improve. Why? Because invariably, if we want to reduce *Touch Time*, we must *change the way work is performed* and *incur costs to perform and sustain the change* and accept the *risk of failure*.

Reducing *Wait Time* does not require changing the way work is performed; it only leads us to make a different start/stop decision; and hence incurs no cost at all, because there are no changes to sustain. The change, if there is one, is the mental adjustment needed to establish the new policies governing such timing decisions - but there are no material or operational changes, because all the work is still done the same way as before.

Changing the way work is performed (acting on *Touch Time*) is always an expensive proposition, because it will require one or more costly actions such as:

- Investments (e.g. purchasing new equipment)
- Staffing (e.g. hiring new people and training them)
- Assets (e.g. buying new offices)
- Restructuring (e.g. putting new management structures in place)
- Reorganization (e.g. moving to new premises)
- Retraining (e.g. getting people up to speed)
- ... and so on!

Not only is it costly, but since the new way of working is unproven, there is also a *huge risk of failure*. Unsurprisingly, most improvement initiatives fail due to resistance to change; but even if the resistance to change is overcome and the risks are mitigated and handled, the benefits from such costly improvement initiatives are negligible if not negative, because in the vast majority of cases the improvement changes are aimed in all places except the only place where they can truly have an impact.

We have also seen that the gain in terms of *Throughput* is (usually) much greater when focusing on *Wait Time* reduction, rather than *Touch Time* reduction. This is a consequence of the fact that in most companies dealing with knowledge-work, *Flow Efficiency* is so bad (typically between 3-7%), that there is not much to gain when targeting the little percentage taken by *Touch Time*.

We might think that focusing on improving *Flow Efficiency* becomes the panacea that will allow ever-increasing improvements. However, there are intrinsic limitations on how much improvement can be achieved this way. Especially when, as in the case of knowledge-work, there is so much VUCA (**V**olatility, **U**ncertainty, **C**omplexity and **A**mbiguity) inherent in the knowledge-discovery processes themselves.

Note that the higher the *Flow Efficiency*, the more predictable and the more deterministic the process is. For instance, highly optimized and robotized manufacturing plants will exhibit higher *Flow Efficiency* than knowledge-work.

In knowledge-work, there are limits to how much can be predicted and determined up-front. The whole point of Lean-Agile thinking is to be able to cope with the uncertainties of knowledge-work; and to adapt to change as it happens. Yet, that should not deter us from striving to improve *Flow Efficiency* as much as it is possible.

It is one thing to accept the limitations on *Flow Efficiency* that are determined by the knowledge domain we are operating in; and it is another to *hunt down the flow inefficiencies which are created by our own ways of working*.

The former (i.e. the intrinsic nature of our domain and *Work Process* - which is literally under our control during *Touch Time*) cannot be eliminated; but the latter (i.e. “how we do stuff around here” - which is how we create *Wait Time*) is simply self-imposed by our own decisions. The inefficiencies of the latter are all self-inflicted and primarily caused by our choice of employing flawed *Mental Models* (which determine when we start and stop work, when and where we invest to improve, and how we consider idle time or excess capacity).

It is obvious: the intrinsic nature of a domain is the same for all competitors in the market - it is the proverbial leveled playing field. While how we decide to perform our work (the “*How we do things around here*”) is where we will win or lose compared to the competition. We can only improve directly where our actions have an effect, within our *Span of Control* (typically while in *Flow Time*), and to a lesser extent inside our *Sphere of Influence* when dealing with customers and suppliers.

We can improve *Flow Efficiency* dramatically by focusing on *Wait Times*. Yet, we must also be aware that there are limits as to how much we can improve this way, an oddity that the *Tameflow Approach* addresses head on. There are only so many common causes that we can address without changing our way of working.

Even if we pursue performance improvements by focusing on *Wait Time* reductions, there comes a point of diminishing returns. A point when any further improvement cannot be achieved unless we actually change the way we work. In other words, *at a certain point you will have to reduce Touch Time strategically* in order to attain any further improvement.

Note: This point of diminishing returns - or of no further returns - is reached when all *Wait Time* is reduced to such an extent that the *Constraint* is working without interruption. Using the TOC terminology, the *Constraint* is “exploited.” At this point, any further gains in terms of *Wait Time* reduction, either upstream or downstream of the *Constraint*, will produce no further improvements - and we will understand why by the end of this chapter.

This brings us to the fundamental question: *where and which Touch Time should we improve?*

For any process which can be broken down into a number of interdependent steps, the improvement effort could be aimed at any one of those steps, to reduce the *Touch Time* expended there. In fact, most popular Kanban implementations will try to improve any one of those steps; as soon as queues or starvation become visually obvious on their boards. The results will be abysmal for all improvement efforts, except for one - which we will find out in a moment.

In most companies, the prevailing *Mental Model* is that the more you improve anywhere the better the resulting performance. In fact, this idea is so widespread that improvement initiatives will be fired at everywhere there is some targetable action in the process. This kind of behaviour only leads to the creation of permanent layers of overhead costs that are permanent in nature, invisible to the analytical eye and irreversible by even the most astute.

Ironically, when one starts effectively improving on *Touch Time* it is still so easy to fail - because there is really only one place in the process where *Touch Time* reduction is worthwhile. This place is the location of the *Constraint* in the process.

Any investment or change aimed at a *non-Constraint* will be totally ineffectual and could even be

counter-productive - i.e. diminish overall performance.

To understand where we should improve, we need to adopt the *Mental Model of Constraints management*.

On Bottlenecks and Constraints

We often refer to “*Constraints Management*” and to the “*Theory of Constraints*.“ The very word “*Constraint*” can be understood in many ways, and it is not always clear what it stands for. So it might be useful to define what we mean by “*Constraint*” here.

It is natural to relate to the word in terms of its common acceptation in the English language. A “constraint” is something that limits our ability to do something. It could be just about *anything*. Like bad weather, if you want to go for a walk. The weather could be so bad that you decide not to take that walk. You can consider that as constraining your actions.

From our more technical perspective, we need a more precise definition:



Constraint: The one thing that prevents us (our entire organization) from delivering more of whatever it is we have to deliver.

So if we build software, then all impediments that limit the production of software is a *Constraint*.

Because *Constraints* are perceived as *limiting*, we might be led to believe that they are *bad*. Yet, for the purpose of our renewed *Mental Model*, we will consider *Constraints* as just being! Knowing where they are good. Not knowing is bad.

Why?

Because the *Constraint* gives us exactly *one* reference and leverage point that we can use to manage the whole system. We can use the *Constraint* to establish a baseline, on top of which we can stack up improvements.

If we had *no Constraints*, we would simply increase our production to infinity. Obviously that cannot happen. Therefore a *Constraint must* always be present; it is part of our physical reality, it is part of nature. The best we can do is to actively manage the *Constraint*.

A *Constraint* gives us an excellent point where we can *focus*. Without a *Constraint*, we would have no guidelines as to where to direct our attention, effort, decision-making or improvement initiatives.

Thinking that *Constraints* are *good* is not the ordinary view of the world. This is a matter of perspective. In our quest to renew our *Mental Models*, we must view *Constraints* as archimedean levers.

So let us start thinking about it in this way: *the Constraint is our best friend!*

Bottlenecks are not Constraints

When looking at a *Kanban Board*, it has become customary to talk about finding *bottlenecks* in the *Work Process* that we employ.

Are “*Constraints*” and “*bottlenecks*” the same thing?

Constraints and bottlenecks are not the same, although many people confuse the two and often use the term “*bottleneck*” when they really mean “*Constraint*,” and vice versa.



Bottleneck: Any kind of resource that cannot deliver up to the demand that is placed on it.

We can have *several* bottlenecks in one process.

To illustrate: if an analyst and a developer are working together, and requirements come in - say - at 100 story points (to use a common Agile sizing metric, just for the sake of argument) per week, and the analyst is able to tackle only 70 story points per week while the developer is able to code only 60 story points per week, then they are *both* bottlenecks. Both are delivering *less* than the demand that is placed on them.

Naturally this raises the question: *what is a Constraint then?*

Let us now refine our understanding of what a *Constraint* is, and learn about the definitions used in *Constraints Management* (i.e. *Theory of Constraints*).



Constraint: The bottleneck (out of possibly many) that has the least *Capacity* or longest *Flow Time*.

We want to identify and pick *this* bottleneck to manage the whole system.

Furthermore, in the *Mental Model* we are trying to build, unless we *deliberately manage* that bottleneck, we will not consider it as a *Constraint*, but just as the bottleneck with the least capacity.

So, even more specifically:



Constraint: the managed bottleneck with least *Capacity* or longest *Flow Time*.

There is no reason to identify the *Constraint* unless we intend to manage it. If we want to manage it, we need to resort to the *Five Focusing Steps* (5FS) which were designed to manage *Constraints*.

In a system where you are managing the *Constraint*, bottlenecks are not a problem; but if you are *not* managing the *Constraint*, then the bottlenecks can make us very confused. We will read off and act on noise from the system, rather than the signals that are really important and matter. Managerial wild goose chases will become the norm.

For example, if the analyst mentioned above is capable of coping with 70 story points per week, but the developer can only deal with 60, and we examine the flow of information or production through the system, the first point where we realize that something is not up to the expectation of demand, is the *first* bottleneck. This will mislead us, because we might start doing an improvement initiative focusing on that first bottleneck. In reality no matter how much we could improve that one, we would not be improving anything at all. (Shortly, we will delve into why this is so.)

The real problem is in the second step, the one that has the least capacity but we could not see that, because our attention was caught on the first bottleneck that was masking the real *Constraint*. Now this was a simple example with only a couple of steps. Imagine how this works out in complex organizations with hundreds or thousands of employees or process steps: the real *Constraint* is almost always masked out by a gazillion bottlenecks!

This negative effect of focusing on what is presumed to be a *Constraint* when in reality it is not, leads to the creation of massive messes.

For example, using *Kanban Boards* with *Column WIP Limits* almost always creates this kind of masking effect, which will make you focus on “ephemeral problems” which are not problems at all.

It is really important to properly identify the *Constraint*.

Note: For a more extensive explanation of why, in particular, *Column WIP Limits* have such negative effects, see *Chapter 18 - Challenges of Work-State Work in Process Limits* in the *Hyper-Productive Knowledge Work Performance* book, where thirteen problems of *Column WIP Limits* are identified and discussed.

Constraints are not only Bottlenecks - There are Other Kinds of Constraints

The idea of managing the *Constraint* is the foundation of *Constraints Management*. It is about thinking deeply about the effects of the *Constraint* as the most limiting factor on the overall performance of the whole system. It is a kind of thinking that can uncover spectacular ways to improve the overall system's performance.

There can be many different types of *Constraints* that reflect the definition of being a limitation to whatever it is you are trying to achieve; and these *Constraints* are *not* bottlenecks at all - and thus we have to rethink all we know about the *Constraint* in terms of the first definition: as anything that mostly limits us to achieve our *Goal*.

Here are a few examples:



Market Constraint: If we are able to produce *more* than what the market is willing to absorb and purchase, the *Constraint* cannot be a bottleneck in our work process, but it is the market itself.

In such a situation, no matter how much we focus on the *Work Process*, we will not increase *Financial Throughput*. Instead, the action should be in terms of marketing and sales, ensuring that there is a demand for our products, and making products that the market wants. Only once there is more demand in the market than we can deliver, should focus return back inwards.



Policy Constraint: Almost all companies have both explicit policies and unwritten rules of "how we do things around here." Such policies and rules can be the real *Constraint*.

The way we think about employees having idle time is often managed by counter productive rules. In the name of *Resource Efficiency*, everybody is expected to "work" or "contribute" and "bear their share" or "be a team player" - but such rules just lead to overproduction and possibly to overburdening of the very *Constraint* we intend to manage. Most of those employees, in fact, will burn out. Pull policies can also have dire consequences on metrics.

Any pull policy that diverges from *First In, First Served* will not reap the benefits of *Little's Law* pertaining to *Flow* and absorption of *Variability*. Class of service and swimlanes on *Kanban Boards* also impair the quality and usefulness of flow data with respect to *Little's Law*.



Management Attention Constraint: Executive management needs to deal with a multitude of issues; to the extent that management attention will be stretched thin and unable to focus on what really matters.

The email inboxes of managers are invariably overloaded. Their calendars are packed with end-to-end meetings. The range, scope and variety of issues they need to deal with is endless. It is no surprise that *management attention* is in short supply; and the situation is even more exacerbated when there is no awareness of *Constraints Management*. If properly introduced, *Constraints Management* can help executives to focus *only* on what matters, and filter away all other non-issues. Whenever management attention is misdirected, it will not only be wasted, but it might also induce managers to make counter-productive decisions. Dr. Goldratt stated that this *Constraint* was one of the most significant.



Mental Model Constraint: The biggest *Constraint* of all is our capacity of developing a *deep understanding* (as Deming would say) of the systems we are managing. In other words, we are constrained by our *Mental Model* of our reality.

This is where having inappropriate *Mental Models* produces severe limitations, and even causes damage. The cost obsessed *Resource Efficiency* policy mentioned above has its origin from a *Mental Model* that leads us to believe that resources necessarily need to be 100% busy, 100% of the time; but this line of thought should apply to the *Constraint* only, and not indiscriminately to all resources.

Where to Improve, Where to Invest

With the insight that a *Constraint* may be any number of things, insisting too much on finding a bottleneck/*Constraint* in the *Work Flow* could divert attention away from identifying the real *Constraint*; thus misleading us and effectively wasting our attention, focus and efforts. However, for the sake of understanding and illustrating what benefits can be derived from focusing on *Constraints*, hereafter - and in the rest of the book - we will effectively concentrate on an internal *Constraint*; one that is in the *Work Process* we want to improve; one that is within our *Span of Control*. To illustrate the case of why it is necessary to focus on the *Constraint*, let's use a simple model. Suppose we have a simple process, as shown in Figure 1.



Figure 1 - Breakdown into operations and their (average) duration.

The process consists of three steps: B, H and G. Under normal circumstances, B takes (on average) 3 days, H takes 5 days and G takes 2 days.

Note: Be aware that this is a *Mental Model*. In the real world of knowledge-work, processes are not as clearly repeatable as in this example. Instead of discrete times, we need to reason in terms of statistical *Flow Time* distributions. However, the logic that we develop with this simplified *Mental Model* still remains valid.

Normally, the common policy of starting work as soon as possible, would bring about a *Cumulative Flow Diagram* that would look as follows:

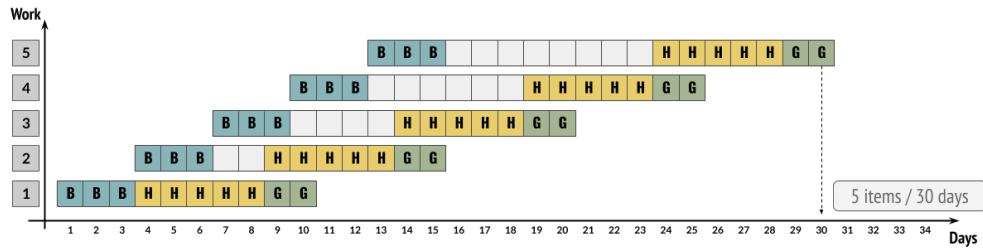


Figure 2 - We start as soon as possible. First operation determines start.

We can see that any *Work Item* has an increasing wait time between step B and step H, simply because H needs more time than B. We also read off the diagram that after 30 days, five items are produced. We can tally 20 Wait days *inside* the process, shown by the empty light gray squares.

Let's suppose we have learned the lesson of removing the in-process *Wait Times* and stacking them before initiating work. Thus we postpone commitment, and we start work at a pace that can be sustained by H. (How this can actually be done, is a topic that will be explored in later chapters.)

Note: H is the *Constraint*, and is effectively determining the *Rate of Delivery*, even if the actual and final delivery step is performed by G.

The *Cumulative Flow Diagram* becomes as follows:

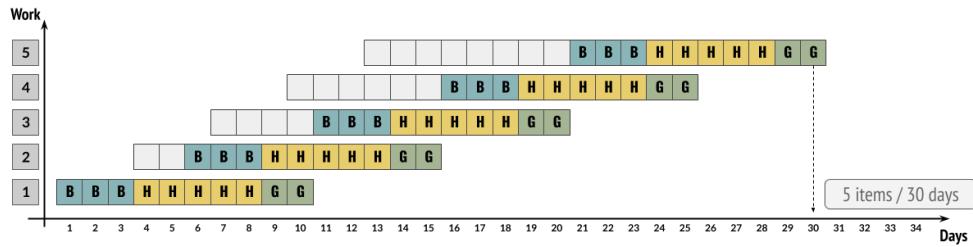


Figure 3. We postpone start. Slowest operation determines start.

Five items are still delivered after 30 days, but we have eliminated the *Wait Time* between B and H. The situation is ideal; because there is no more *Wait Time* to weed out inside the process. *Flow Efficiency* is 100% as there is not a single empty light gray square from the time we start to the time we finish.

Remember again: this is a *model* - that allows us to *think* about an ideal scenario! Naturally, we know this scenario is impossible in reality (especially in knowledge-work). Yet, this gives us a *Mental Model* about what we could do to improve performance once there is no more *Wait Time* to squeeze out of the process. Naturally, at this point, with no more *Wait Time* we can only improve by attacking the *Touch Time*.

See! The model forces us to reason about the impact of touching, well, the *Touch Time*. We know that it will require investments, changes and carry some risks; so we need to be very considerate about what we are about to do.

Suppose we ask two consultants, David and Steve, to come up with a proposal, and we are presented with two alternatives, illustrated as follows:

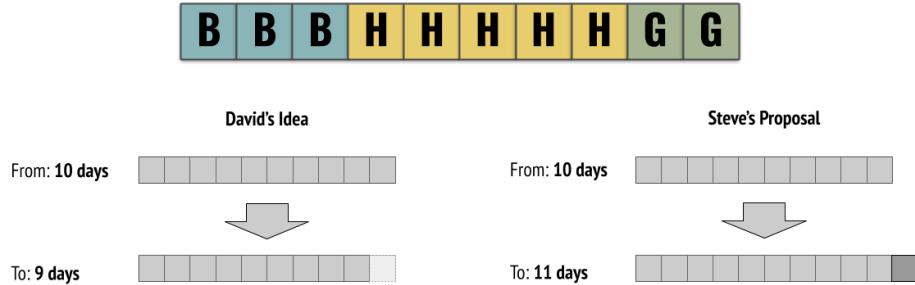


Figure 4. Two proposed changes. Which one is best?

David's proposal will *reduce* the end-to-end *Flow Time* of the process by one day bringing it down to 9 days. The consultant insists that reducing "lead time," as he is fond of saying, will bring great benefits. Suppose further that the investment needed to sustain this change is \$20,000.

Steve's proposal will *increase* the end-to-end *Flow Time* of the process by one day, bringing it up to 11 days; and the investment needed is \$30,000.

Which option should we choose?

Most companies would not hesitate to go for David's proposal; and quickly dismiss Steve's as unreasonable. Not only does it increase the end to end time of the process, but it even costs more - the cost accountants would dismiss it with no hesitation, not even looking at the ultimate impact on the overall performance.

But is it really so foolish?

Are not all *Lean* initiatives about reducing *Flow Time* (or "Lead Time" as they call it)? And here we are presented with an option, Steve's, that not only increases *Flow Time* but actually costs more than David's, which decreases *Flow Time* and costs less.

To be able to decide which alternative brings the most *Return on Investment*, we need to figure out the impact of the changes in the *Financial Throughput* of the system, rather than blindly focus on the reduction of *Flow Times*.

After a deeper investigation we discover the following:

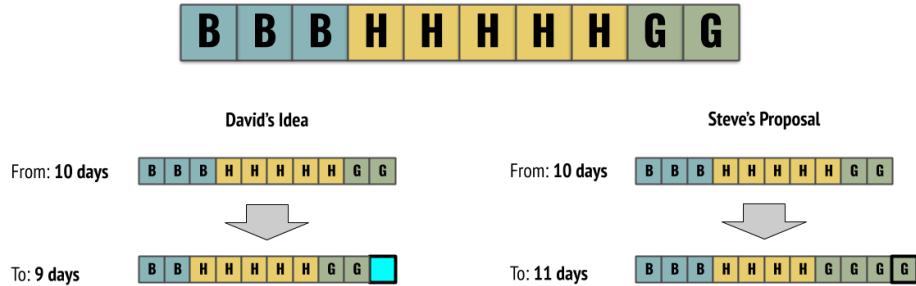


Figure 5 - Process effects of the two changes.

David's improvement proposal happens by reducing B's *Touch Time* by one day.

This has to be compared with Steve's improvement proposal which reduces H's *Touch Time* by one day, and at the same time also increases G's *Touch Time* by two days.

So the two proposals act on two very different places of the overall *Work Process*.

Let us first depict how David's proposal stands out to the original situation (illustrated in figure 3). The two *Cumulative Flow Diagrams* look as follows:

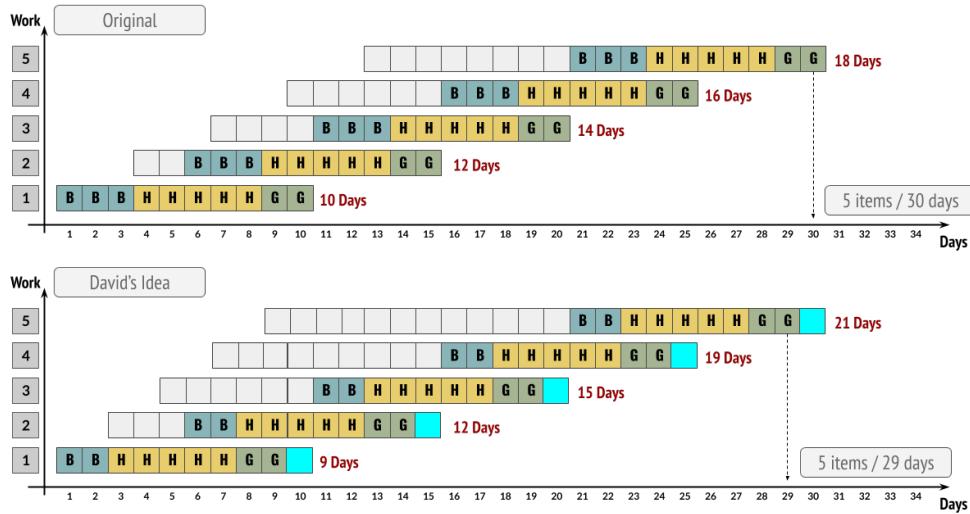


Figure 6. First change scenario (“David’s Idea”)

We can see that David's idea does indeed produce an improvement. Five items are delivered after 29 days rather than 30. The improvement is hardly spectacular. In fact by looking at the diagram, we can conclude that any item will be delivered *just one day earlier* compared to the original situation.

And that's it.

There is no cumulative effect and the lifetime benefits are consumed after the first 10 days.

Furthermore, the diagram also shows that if the original policy of starting work as soon as possible were still in place, the situation would definitely appear to have degraded, notwithstanding the improvement. Consider the fifth item: in the original setting (again with the original policy!) it took 18 days from start to finish; whilst with the “improved process” it would take 21 days from start to finish.

Puzzling, is it not?

The insight here is that if you do not keep in mind the *Mental Models of Flow Efficiency* - and are stuck with the original policy of starting work as soon as possible, and improve just anywhere it seems to make sense - then this “process improvement” would make the situation *worse* because all work would be started increasingly sooner, but delivery would gain only a constant of one single day.

With this in mind, it is not surprising that many “process improvement” initiatives result in failures, notwithstanding that they might improve *Resource Efficiency* or *Process Efficiency*.

Now let us compare Steve's proposal to the original situation; it would look as follows on the *Cumulative Flow Diagram*:

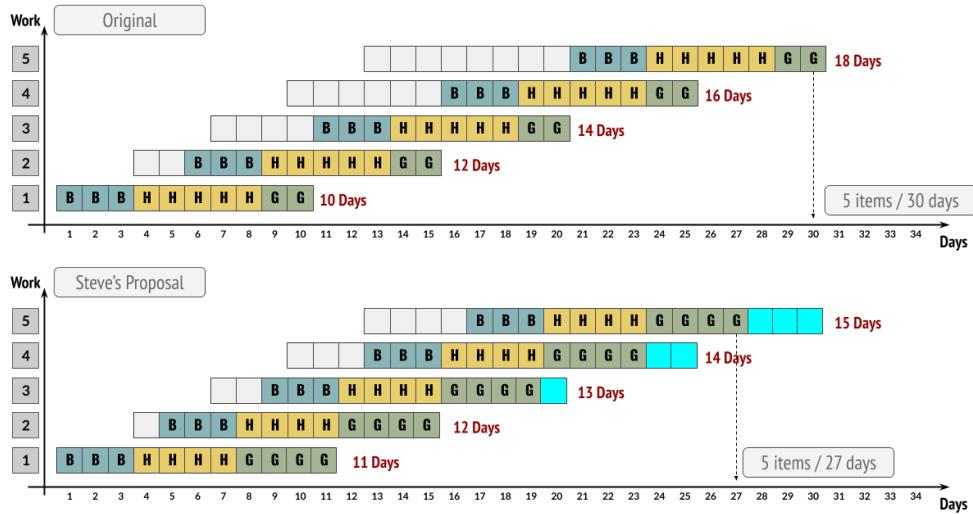


Figure 7. Second change scenario ("Steve's proposal")

Remember that Steve's improvement proposal shaves off one day on H's *Touch Time* and adds 2 days to G's *Touch Time*.

Now the situation is interesting. The end-to-end *Flow Time* is longer – but we observe something unexpected, even surprising: five items are delivered after 27 days rather than the original 30 day – and rather than 29 days as in David's proposal!

The first item is, comparatively, one day late. The second item is delivered on the same day in both cases. From the third item onwards we see the pattern: *Steve's improvement proposal is cumulative and gains one day for every additional item.*

If we take into consideration the original policy of starting as soon as possible, Steve's proposal is clearly an improvement too. Let us again consider the fifth item: in the original setting (again with the original policy) it took 18 days from start to finish; whilst with this "improved process" it would take 15 days from start to finish.

In fact, every item would be delivered increasingly sooner. The insight here is that even if you do not keep in mind the *Mental Models of Flow Efficiency*, and are stuck with the original policy of starting work as soon as possible, then Steve's proposal would actually make the situation better, because all the work would be started at the same rate as before, but delivery would still gain one day for every item after the second one.

Finally, let us compare directly David's proposal to Steve's.

Here are the two *Cumulative Flow Diagrams* diagrams:

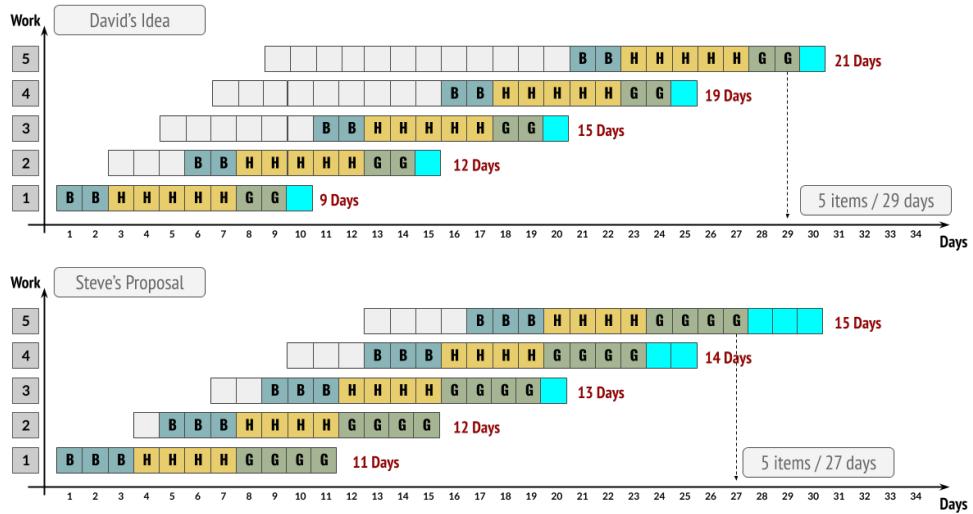


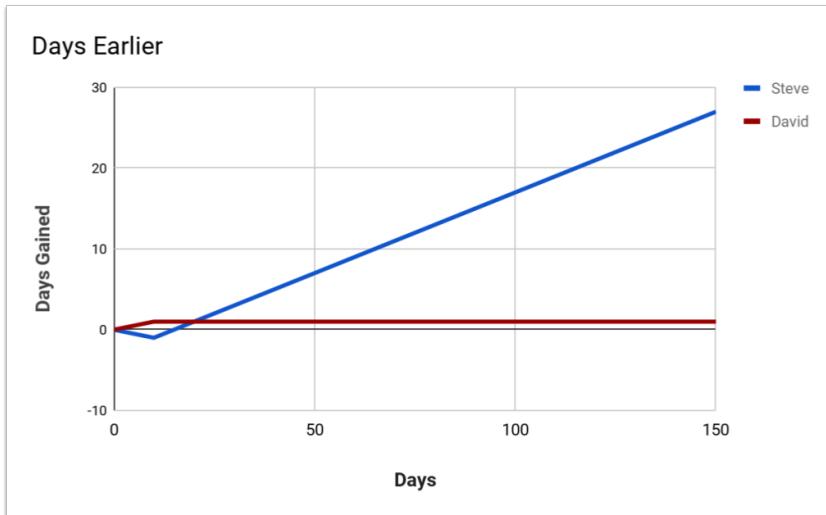
Figure 8. The two options compared directly.

The improvement is visible in the cyan colored days at the end of the process: they represent the overall time reduction. The essential difference is this:

- David's diagram is simply “left-shifting” the delivery line by one day.
- Steve's diagram is “increasing the slope” of the delivery line, and it will gain one day for every item beyond the second one.

The lasting impact of David's idea is nil. Once in place, step H still controls the *Rate of Delivery* in the same way as before. The pace of David's idea is still one piece every five days; whereas with Steve's proposal the pace becomes one piece every four days.

To illustrate this even more clearly, let us focus on how many days earlier work is delivered, in comparison to the original timings:



The impact is dramatic. After 150 days, David's proposal would have gained only one day with respect to the original timing; while Steve's proposal would have gained 27 days. And the trend will continue: David's proposal would never gain more than that first single day, while Steve's proposal would continue to gain one day every five days forever.

The cost differential of \$10,000 in favor of David's proposal is a mirage and was all about accounting fiction. The additional sales generated by Steve will never cease to generate *Financial Throughput* - a topic that will be explored in the next chapters.

The real difference is that David produced the improvement on the *Touch Time* of a *Non-Constraint* (B), while Steve produced the improvement on the *Touch Time* of the *Constraint* (H).

We do not look into the details of exactly how either David or Steve produced *Touch Time* reduction. We know for sure that if they succeeded with that reduction, then there must be *some* change in "*how things are done*"; and therefore, there must be *some* investment that must be sustained. (This is unlike the case when we are pursuing *Wait Time* reduction, where there is no operational change at all; but only the adoption of different decision-making criteria as to when to start work.)

What we need to realize is that David's improvement targets a *Non-Constraint*, while Steve focuses on the *Constraint*. Also notice that Steve's solution involves step G as well: there is investment expended on changing how G works.

Now while G is not a *Constraint*, the change is necessary in order to improve how H is performing. Effectively, this is an illustration of a "*subordination*" change, which is the third step in the *Five Focusing Steps (5FS)* of the *Theory of Constraints*.

Special attention must be given to this: it is not that we never invest in changes upstream or downstream on *Non-Constraints*. We do so, but only if we are perfectly aware of why; and specifically that by investing on selected *Non-Constraints*, we are diminishing the *Work Load* on the *Constraint*.

In the case of G, we want G to subordinate to H; but at the same time we produce a change for H, so that H can do *less work* in order to allow the *entire system to deliver more*.

Step H was the *Constraint* in the original process, because it had the longest *Touch Time* (5 days) of all

steps involved. If we look at the two proposals but focus only on the change that affects the *Constraint*, we observe the following:

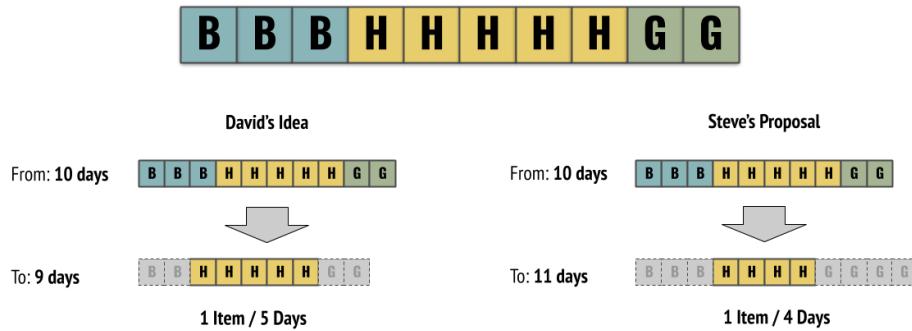


Figure 9. How the changes affect the Flow Time on the Constraint.

David's improvement does not reduce the *Touch Time* on the *Constraint* at all. All it achieves is a one day gain but *only on the first item when it is touched the first time by B*; and that's it. Thereafter, the process has the same performance as the original one. No matter how many items are produced, the entire gain would be that one single day once - but the recurring costs of maintaining the improvement will last a long, long time! The original process delivered an item every five days; the improved process still delivers an item every five days (just left-shifted by one day, with respect to the original one). Notice that *Throughput* is still $1/5 = 0.20$ items per day, the same as in the original process.

In contrast, Steve's improvement affects the *Touch Time* of the *Constraint* at H. It is decreased from 5 days to 4 days. This means that the improved *Throughput* is $1/4 = 0.25$ items per day. As it were, Steve's improvement would produce an increase of 25% in *Throughput*!

Notice that in this model, the impact on *Flow Efficiency* between David's and Steve's proposals is the same because 100% of work is postponed with the same strategy in mind. So there is no *Flow Efficiency* difference causing the different outcomes.

Note: This does not mean that we have run into a limitation of the *Flow Efficiency* metric. Remember we are illustrating a *Mental Model* where even if we look at the effectiveness of the *Constraint*, the *Constraint* is always 100% “busy” in both examples. The example is construed on purpose like this in order to force us to reflect on where to focus improvements once *Flow Efficiency* has already been

resorted to. In the real world, you will always resort to a mix of both, and typically first by improving *Flow Efficiency*, and then by acting on the *Touch Time* (or *Process Efficiency*) of the *Constraint*.

It is interesting to note that when the *Touch Time* improvement does not affect the *Constraint*, then it produces just a mere left shift of the delivery line and a new layer of overhead costs. However, since we know that *Flow Efficiency* is very low for knowledge-work, the impact of improving the *Touch Time* on a *Non-Constraint* is negligible.

Note: A deeper insight here is that *Little's Law* can be refined: it should be expressed only in terms of the *Flow Time* at the *Constraint*. Reducing *Flow Time* on *non-Constraints* produces a mere and minor “left shifting” of the *Rate of Delivery* line, but not an increase of its slope (i.e. increase in *Operational Throughput*).

In fact, the result will most likely be negative, because of the extra costs in terms of *Operational Expenses* and/or *Investments* needed to sustain the change. We are likely to end up with a permanent and invisible increase in overhead costs: that is a real “waste” - as we call it.

On the other hand, when improvement actually affects the *Touch Time* of the *Constraint*, then the slope of the delivery line will increase, resulting in a real and lasting increase in *Throughput*.

Takeaways

In summary, so far we have learned that to produce an effective performance improvement, we need to do the following:

1. Limit *Work in Process* and *Postpone Commitment*.
2. Reduce *Wait Times* (inside the process) and improve *Flow Efficiency* without changing the *Work Processes*, nor sustaining any *Investments* nor higher *Operating Expenses*. The risk of failure is practically non-existent, because the only thing that changes is the decision about when to start new work.
3. Reduce *Touch Time* only if you are absolutely sure that you are doing so on the *Constraint* of the system. Reducing *Touch Time* means operating in new ways. The *Work Processes* will have to change. You will have to sustain both initial investments as well as higher operating expenses. If you reduce *Touch Time* on any *Non-Constraint*, the effect will be negligible and most likely negative. (And the only exception is if that reduction is done to support the improvement on the *Constraint*.)

The point here is that only Goldratt’s *Theory of Constraints* and the *Tameflow Approach* focus on improvements at the *Constraint*.

Flow Metrics are simple yet extremely powerful managerial tools when leveraged on our company’s *Constraint*. They are also symmetrical, transposable and transportable as their mastery comes at no cost when moving from one project to the next or to a new company. (This is in stark contradiction of context

specific Agile environments where ramp up costs are substantial due to the infinite numbers of practices that Agile theoretically supports. No wonder Agile is fragile to team member turnover.)

One has to wonder why enterprises adopt frameworks such as Scrum or SAFe in the hope of achieving commonality of purpose and practice, or in an attempt to reassure themselves that they are on the right path. Lack of Agility - or worse Agility through obscurity brought about by new artefacts, roles, responsibilities, structures and ceremonies - will only lead us to more bureaucracy. It is that kind of obscurity that leads Agile to wanting to keep teams intact and to totally ignore the Agile Manifesto, which never demanded a single new role, extra responsibility, team structure, additional artefact or one more ceremony.

Flow Metrics do not require us to keep ceremonies while role playing in theatrical agility. They are a lens that allows us to zoom in on to the *Constraint*. They don't impose anything, but enable us to *think about* and *figure out* the best thing to do from the perspective of the *whole organization*.

Robert Newbold says...

This book contains information every manager needs to understand.

Robert Newbold, CEO at ProChain Solutions, Inc.; former Director of Software Development, The Goldratt Institute; author of *“The Project Manifesto: Transforming Your Life and Work with Critical Chain Values.”*

Corey Ladas says...

I applaud the approach of grounding everything thoroughly in the underlying theory. This book has the potential to bootstrap a correct method.

Corey Ladas, Author of *“Scrumban: Essays on Kanban Systems for Lean Software Development,”* inventor of Scrumban, Personal Kanban and some of the practices that are associated with the Kanban Method.

PART 3—Making Accounting Make Money

The ability to generate cash is what keeps businesses alive. The conventional *Mental Models* we have grown to adopt in making managerial decisions have little concerns for growth. When the time comes to make more money, traditional *Cost Accounting* has a reductionist perspective that has deep roots in the fabric of society: slash costs or produce at maximum capacity to get to the lowest product cost and draining the firm's most valuable asset in so doing: its cash! All the while alienating their most vital components: its people!

Can we make better decisions faster? Definitely, if we accept that there is an avenue for growth.

But how do we get there? How can we make better decisions? This is where common improvement models fail us.

Global systems measurements is what is required. The model is so simple that we only need to understand only three levers: *Throughput*, *Operating Expenses* and *Investments*!

Note: This Part stands out from the rest of the book because it focuses primarily on the topic of *Financial Throughput* rather than *Operational Throughput*. It is also a tribute to Dr. Goldratt and his business novel '*The Goal*', with the narrative developing through dialogues. Without *The Goal* this book would not exist.

Bob Sproull says...

This is one of the best books I have ever had the privilege of reading and I mean that! Just when I thought that one chapter was the best in the book, the next chapter took its place. I will be re-reading this book again and again and will advise everyone that I know to purchase this gem.

This book will become a classic.

Bob Sproull, Owner at Focus and Leverage Consulting; Lean Six Sigma Master Black Belt; TOC Jonah; author of “Epiphanized, A Novel on Unifying Theory of Constraints, Lean, and Six Sigma,” “Focus and Leverage: The Critical Methodology for Theory of Constraints, Lean, and Six Sigma (TLS),” and “Theory of Constraints, Lean, and Six Sigma Improvement Methodology: Making the Case for Integration.”

Russell Eirich says...

I am loving this book. It is pretty big. But I have to admit, it is great and each and every page is highlight worthy.

Russell Eirich, Senior Scrum Master/Coach (CSP), Scrum Management Office, Health Learning, Research & Practice, Wolters Kluwer.

6—Introduction to *Throughput Accounting* and Culture

There should never be a discussion pertaining to culture in organisations without a deep understanding of Dr. Goldratt's *Throughput Accounting* and its impact toward alignment, motivation, cohesiveness, unanimity-based decision-making, behavior - both at the individual and group level - and social engineering.



Throughput Accounting (TA): "A management accounting method that is based on the belief that because every system has a Constraint which limits global performance, the most effective way to evaluate the impact that any proposed action will have on the system as a whole is to look at the expected changes in the global measures of Throughput, Investments and Operating Expenses."

(Source: TOICO Dictionary, Sullivan, 2012).

In the *Tameflow Approach*, the definition of *Lead Time* is linked to cash and appropriately called *Order to Cash Lead Time*. It is the time from when a customer places an order until the time the customer has paid fully. This is a lesson that goes back to Taichi Ohno: "All we are doing is looking at the time line from the moment the customer gives us an order to the point when we collect the cash. And we are reducing that time line" (in his *Toyota Production System: beyond large-scale production* book, 1988).

Notice that if *Lead Time* is defined more liberally as "Time to Market" or "Order Lead Time" and does not include the final payment (as is the case in the *Kanban Method*), then there is no drive to improve the overall systems with respect to financial performance.

In accounting terms, *Lead Time* as used in *Throughput Accounting*, follows the order to cash cycle and thus works on a cash basis. "*Lead Time*" as defined in the *Kanban Method* is more in line with accrual accounting. Notably, the *Kanban Method* is phlegmatic to the timing difference in collecting cash, making "*Lead Time metrics*" looking much better than what they are in reality.



Accrual Accounting : An Accounting method that records revenues and expenses when they are incurred, regardless of when cash is exchanged.

Throughput Accounting certainly adopts the attitude that collecting cash is under our *Sphere of Influence*, and we should be proactive to follow up on actually collecting the cash and improving performance in so doing.

TA is based on *Flow*. *Flow* is not static and henceforth, every financial analysis must be taken from the viewpoint of the differential stance from the current situation.

Kanban Models and Culture

TA is used as a pedagogical model to explain and illustrate *Flow* in the *TameFlow Approach*, with respect to *Financial Throughput*. *Little's Law* (LL) is used in the same manner, with respect to *Operational Throughput*.

Interestingly enough, both TA & LL are congruent and drive any metrics-based improvement initiative toward the same direction: they will guide you to increase *Throughput*: whether *Financial Throughput* (TA) or *Operational Throughput* (LL). The positive improvement in one metric will necessarily be reflected in the other.

Note: In the *TameFlow Approach* the driver for *Operational Throughput* improvement is the very notion of *Constraints* management. This is entirely consistent with LL, as we have seen in the earlier chapters. The equations of LL become even a more powerful lens when they highlight the impact of improving on the *Constraint*.

It is now time for a story about *Throughput Accounting* and ... culture! What is about to be revealed always comes as a shock to Agile coaches who call themselves transformation experts and culture-centric change agents. Culture is a reflection of the company's systems; and the company's systems are a reflection of the decision makers' *Mental Models*. *Cost Accounting* is the most inescapable system insofar as the economics of the firm is concerned, so if we want to have any chance of creating an impact, we need to address the accounting system.



Cost Accounting: A method of collecting, analysing, summarizing and evaluating various alternative courses of action. *Cost Accounting* supports the *Generally Accepted Accounting Principles* (GAAP) to satisfy outside reporting requirements; though it is also used to make management decisions.

Agile coaches that cannot pass a basic TOC/TA test, will not likely be able to change culture anywhere. If competent, they will use TA as the weapon of choice and will succeed.

Accounting Conflicts

At some point, we need to understand that the bottom-line counts! To that end, we must be literate enough in both *Cost Accounting* (CA) and *Throughput Accounting* (TA) to influence top management to rewire their day-to-day perspectives toward performance management.

CA and TA always reach different - actually opposing - conclusions when it comes to daily managerial action and behavior. The way we think and come to conclusions with our systems has a major impact on culture.

Fortunately, TA and CA are fully reconcilable with the proper accounting adjustments under the period cost accounts. So the translation of financial information is not that difficult of a chore. CA is required for *Financial Accounting* and regulatory requirements, while TA should always be used for financial performance and daily decision-making.



Financial Accounting: the processes and practices of preparation of financial statements (reports) for the owners and external stakeholders.

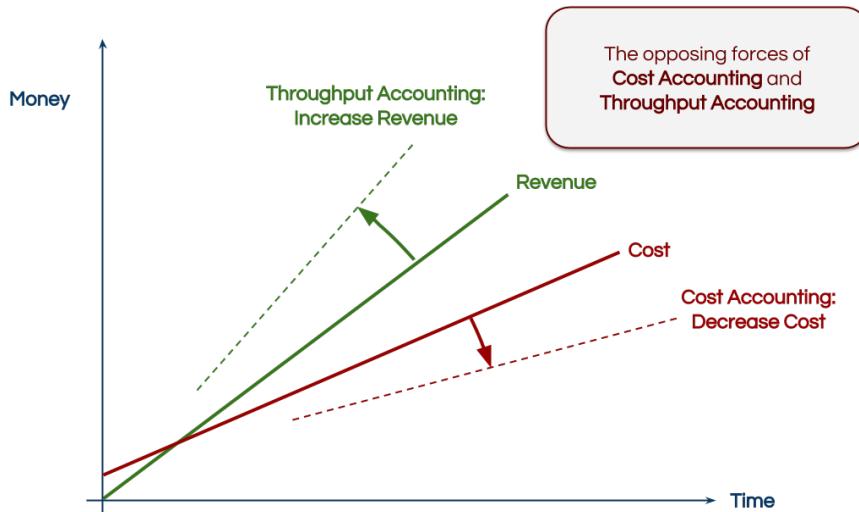
Financial Accounting has little use for management of daily activities and operational decision-making.

For the sake of succinctness, suffice to say that CA assumes that all costs are variable and allocates them in relation to some activity, cost driver or volume. In so doing, CA creates a distortion of reality at the product costing level; a distortion so troublesome that it hinders fruitful managerial decision-making!

We have entered the world of *accounting f(r)iction*.

In the construct of CA - and the use of its data at the operational level - the link between decisions, actions and results is delayed, non linear and without causality. This is totally the opposite of TA where cost allocation is forbidden and where focus on the *Constraint* allows for sharp decision-making that has immediate impact on the financial health of operations.

The profit geometry of CA and TA shown below is also noteworthy. It illustrates the need for cost-cutting for the former and revenue growth incentive for the latter. Naturally TA is all about managing the *Constraint* of the system according to Dr. Goldratt's *Theory of Constraints* (TOC).

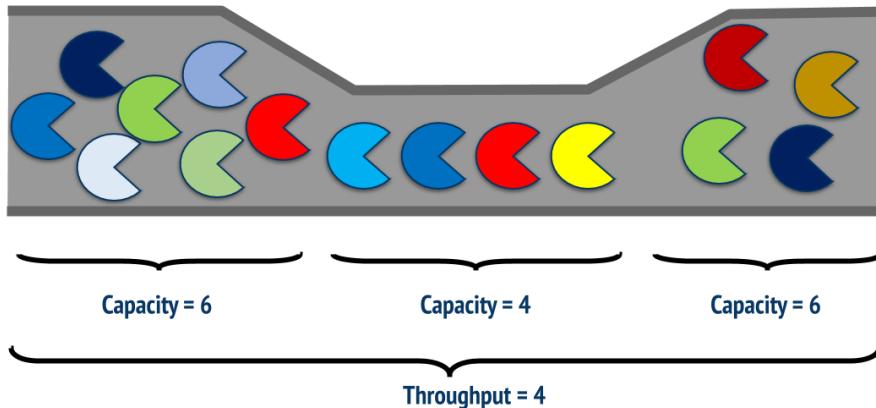


We see by the geometry that managerial choices are pretty evident. Cost cutting is the low hanging fruit in the CA world. In the TA perspective, the slope of the revenue function with regard to the cost function motivates one to increase *Financial Throughput*.

Note: By *NOT* focusing on the *Constraint*, the monies injected into ADKAR (Awareness, Desire, Knowledge, Ability and Reinforcement), KMM (Kanban Maturity Model), CMMI (Capability Maturity Model Integration), Scrum, SAFe (Scaled Agile Framework), and all other such improvement initiatives go directly toward increasing the slope of the cost line and adding to the existing layers of overhead costs in a permanent, invisible and uncontrollable manner - without actually doing anything to deliberately increase the revenue line.

Smart Money at the *Constraint*

Selling TA is always a challenge as it requires one to change *Mental Models*. The most powerful illustration to make the case appears below where traffic flows from left to right:



We can clearly see the *Constraint* of the system where *Capacity* is at its lowest at four. If given 2 million dollars to improve the system, would we choose the leftmost or rightmost boundary where our money would go to waste toward improving the system's overall *Throughput*?

The *Capacity* at the *Constraint* is the limiting factor and it is located in the middle in this example. Putting money anywhere else but at the *Constraint* is a bad decision.

With this traffic jam example we now have an intuitive understanding of what "*Throughput Accounting*" is in the physical world: put all improvement efforts on the *Constraint* or increase overhead costs permanently. It is as simple as that.

Unanimity or Consensus Based decision-making?

Let us revisit culture again. Have you noticed in this simple example how unanimity-based decision-making can be achieved once the team knows where the *Constraint* is in the traffic jam illustrated above? Improvement efforts with the most direct and significant impacts on *Flow* must take place at the *Constraint* where the *Capacity* is the lowest at 4 and nowhere else.

Will anyone support or propose going on a wild goose chase away from the *Constraint* again? Politics and irrationalities are easier to take out. Procedural justice is built into TA.

This is a major change from the consensus-driven decision-making *Mental Models* that the Agile (and in particular Scrum) movement promotes and cherishes. It drains people's energies, makes everybody

unhappy and has to be (re-)built and sustained continuously. There are of course circumstances where consensus based decision-making is warranted, but now that we understand how focusing on the *Constraint* facilitates unanimity-based decision-making, consider the bad impact on coordination costs, transaction costs and motivation involved with building consensus ad nauseam.

Now, imagine the benefits of unanimity-based decision in relation to alignment, strategy, morale, empathy, cohesion, and respect. We are now talking culture here, not *Throughput Accounting* any longer!

Throughput Accounting Basics

Let us do a little decision-making drill by only mastering four simple variables of *Throughput Accounting* : *Financial Throughput*, *Operational Throughput*, *Operating Expenses* and *Investment*!

The definitions that follow are derived from those of Etienne Du Plooy:



Financial Throughput (TH) = Revenue - Totally Variable Expenses (TVE).



Operational Throughput (TP) = The rate (or “speed”) at which goods or services are delivered to the market.



Investment (I) = Investment is a global measure and includes all the money in the system such as equipment, fixtures, buildings, tangible and intangible assets etc. that the system owns.



Operating Expenses (OE) = All expenses that do not fit into (I) and (TH) and that contribute to the conversion of (I) into (TH) from an operating perspective.

Pretty simple and intuitive.

The only key thing to remember is that costs that do not have a correlation coefficient of 1.0 with *Financial Throughput* are uncoupled from revenues and end up in the *Operating Expenses* or *Investment* accounts.



Totally Variable Expenses (TVE): Costs that have a 1.0 correlation coefficient with *Financial Throughput*.

Insofar as daily decision-making goes, we ask ourselves, in order, if we have:

1. Increased *Financial Throughput*
2. Decreased *Investments*

3. Decreased *Operational Expenses*

All improvement initiatives that focus on *Constraints Management* benefit from the decoupling of a direct relationship between costs and revenues. This might seem inconsequential at first but makes visually obvious the increasing divergence of the cost and revenue curves in the short term. The visibly apparent separation in the slope of the cost and revenue functions is a clear signal of profitability occurring and that the ongoing improvement process is forging ahead.

Takeaways

There is a fascinating accounting story and culture lesson to be learned here for knowledge-workers. Once the *Constraint* is made visible to all, magic happens! Debates are no longer necessary as to what makes better sense, decision-making is faster, stress goes down. Collaboration, empathy, new behaviors and improved attitudes rise.

There is this trendy line in the Agile Coaching world and it goes like this: “*Culture eats strategy for breakfast.*” Popular delusions attract crowds and make us feel good, set us up for inertia as little actionable guidance is ever provided that goes directly, promptly and straight to the bottom line.

How do we realize this? With *Throughput Accounting*!

The *TameFlow Approach* is all about creating the conditions wherein a performance focused *humane culture* can thrive, rather than trying to fit a new square culture peg into an old round accounting practice hole, without addressing the core conflicts caused by the *Cost Accounting* mindset that produces such dysfunctions in the first place.

To address HR concerns, *Throughput Accounting* is easy to master. It works at the product level, the team level, the division level, the project/portfolio level and the enterprise level. It can be understood by accountants, engineers, clerks, developers, PMs, HR personnel and so on.

Throughput Accounting weeds out complexity and puts causality in with regard to your daily decision-making so that you can achieve superior performance. The best time to plant a tree was 10 years ago. The next best time is today. It is not too late to pivot away from improvement models and Agile approaches that focus away from the *Constraint* and that have no tangible impacts on the bottom line, and even less on the organization’s culture.

Johanna Rothman says...

If you're trying to create an agile culture, do yourself a favor and read "Tame your Work Flow." From the explicit mental models to understanding how flow works in an organization, to the whole issue of accounting and money, this book helps you understand what you might - and might not - do to create your business agility culture.

I particularly liked the comparisons between cost accounting thinking and flow accounting thinking.

From project teams to multiple teams to managers, "Tame your Work Flow" offers plenty of ways to rethink how you work.

Johanna Rothman, Consultant and Writer

7—Accounting F(r)iction

The Story of Daniel at TRUSTNEWS

TRUSTNEWS (a fictitious company) is offering a Content Management System (CMS) for the news publishing industry. Its mission is to provide industry leading, subscriber financed SaaS (Software as a Service) solutions on the cloud to companies that distribute news content worldwide such as Flash News, 7/24/365 - both fictitious companies as well - and the likes.

In this fast market, ingenuity is of the essence. New functionality is going to push TRUSTNEWS ahead of the pack. The existing development platform is Open Source based and industry expertise was the battlefield in which TRUSTNEWS differentiated itself with software engineering excellence and delivery. It was necessary at this point to improve operations and the internal processes of the software development cycle.

Let us meet Daniel who has been given the task of improving TRUSTNEWS' software engineering *Throughput Capacity*. Daniel is the agile champion and a big aficionado of SAFe and *Cost of Delay* (CoD).



Cost of Delay (CoD): A measurement of urgency expressed in terms of loss of value over time. Typically the value is denoted in money; but it can also be described with other quantitative parameters or even in qualitative or subjective terms. The intent of CoD is to give a sense of how the value decays as time passes. It can be considered as the cost of allowing a period of time to pass without realizing the value.

Daniel runs the software engineering department and has been waiting for this opportunity for a while. He will also take advantage of the occasion to improve the quality of life at work. This undertaking would mobilize one quarter of the software engineers he had at his disposal. It was significant in scope.

He has now narrowed down the pack of alternatives to two : Option A and Option B. The figures for capacity and the statistics for both options were the results of a study.

Now Daniel has to make the business case and cost the project for approval purposes. He will do it his way. He has no budget but could leverage TRUSTNEWS' internal resources and expertise.

Cost Control and Flow Accounting at TRUSTNEWS

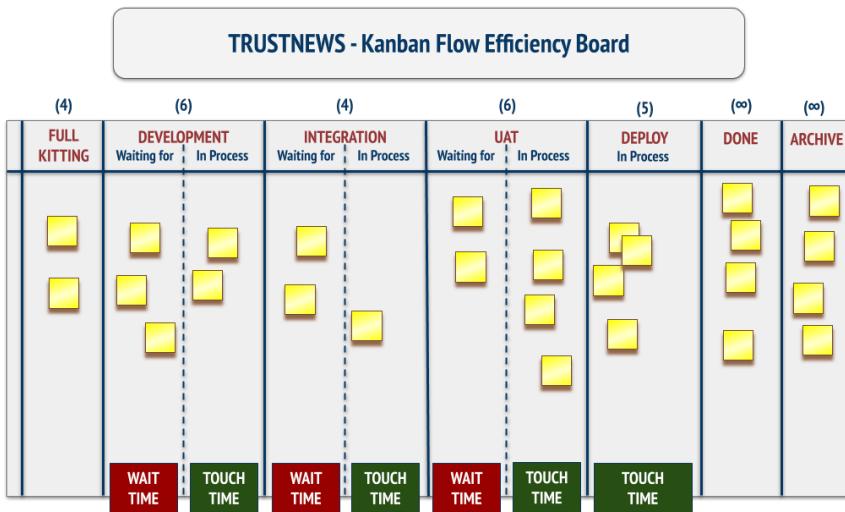
The CEO has urged Daniel to use the massive amount of costing data in the *SCAM - Special Cost Accounting Module* - of the enterprise ERP but Daniel has “declined” the offer. The CEO, who has a good rapport with Daniel, jokingly said : *“No wonder, coming from a PM who only puts the cost centers and budget accounts in his MS Project files.”*

The CEO liked to call Daniel the “one liner PM” because his all-purpose project plan was only used for payroll and budgetary/cost control reasons, and only had a few lines for data entry. It drove both the PMO and the entire accounting department on the third floor bonkers!

The amount of time saved and the overburdening of knowledge-workers to keep those numerous timesheet systems (payroll, project, activity, time management etc.) in a pristine state were two of the reasons people have always wanted to work on Daniel's projects that were all laid out on the *Kanban Boards* on the walls of the fifth floor: the Executive floor no less!

When Daniel joined TRUSTNEWS, the *Flow Efficiency* of the entire engineering department in delivering features was a mere 3%, with favorable wind conditions! Half a year after his decision to implement the *Flow Efficiency Kanban Board* shown below, the motivation and psychological engagement of the teams increased. The cost of delivering software features to enhance the CMS decreased significantly.

Flow Efficiency now stood at 25%. The impact on productivity gave Daniel tremendous political capital vis-à-vis the CEO.



Note: For most readers, terms such as *Full-Kitting* and the appearance of the TameFlow boards in this chapter might be novelties. Their mode of functioning are explained in this chapter and in *Chapter 14 - Flow Efficiency, DBR and Tameflow Kanban Boards*. *Full-Kitting* is addressed in *Chapter 17 - Introduction to Full-Kitting* and *Chapter 18 - Full-Kitting as an ongoing Executive Activity*.

After having climbed a few pay grades since his arrival, Daniel is now worried. *Flow Time* performance has decayed in the last weeks and *Flow Efficiency* has plummeted to 20%. He does not know why it is going down. Has the system changed in ways unbeknownst to him? He just can't figure this one out and it is keeping him awake at night.

Coming back to the cost analysis of Option A and Option B, the first task at hand was to deal with the new anticipated overall *Flow Times* in hours per feature as they would be used as the denominator for his *Cost of Delay* calculations. Both Option A and B would simply leverage the existing expertise of the highly skilled workforce consisting of consultants and full time employees. The focus of each project within the value stream was altogether different however, as illustrated below.

Option A is bringing average *Flow Time* down one hour to nine hours per feature, and Option B is increasing average *Flow Time* to eleven hours per feature as illustrated by the table below.

| | Current | Option A | Option B |
|--------------------------------|---------|----------|----------|
| Flow time in hours per feature | 10 | 9 | 11 |

Daniel had considered the two alternatives because each addressed automation and operational issues that preoccupied him. The details can be appreciated in the next table. Option A shaved one hour off the 10 hours cycle by decreasing process time at DEV and UAT by one hour each and increasing the process time at INTEGRATION by one hour per feature.

Option B decreased INTEGRATION process time per feature by one hour while increasing DEV and UAT by one hour each.

The choice was pretty obvious for Daniel. The break down of each scenario appears in the following table:

| | Current | Option A | Option B |
|--------------------|---------|----------|----------|
| Total Flow Time | 10 | 9 | 11 |
| Flow at Constraint | | | |
| Development | 2 | 1 | 3 |
| Integration | 5 | 6 | 4 |
| UAT | 2 | 1 | 3 |
| Deploy | 1 | 1 | 1 |

But to make sure, Daniel would include the bottom line impact on *Operational Throughput* and see how many additional features per year could be brought about by each option.

As we can see in the table below, TRUSTNEWS operates at a capacity of 5,840 hours a year. The table clearly shows that Option A is more attractive as it can squeeze in 65 more features per year. Option B is to be rejected as it cannot even produce the current output of 584 features per year.

Operational Throughput (TP) - Features Accounting Report - Option A and Option B based on overall average Flow Time

| | Current | Option A | Option B |
|---------------------------|---------|----------|----------|
| Capacity (hours per year) | 5840 | 5840 | 5840 |
| Average Flow Time | 10 | 9 | 11 |
| New features per year | 584 | 649 | 531 |
| Variance | NA | 65 | -53 |

Daniel was pleased with the results of his analysis. Using the average *Flow Time* of his process of 10 hours as a baseline denominator was similar to using duration in *Cost of Delay* (CoD) calculations. He was comfortable with Option A providing a reduced overall *Flow Time* of 9. It was the right thing to do, but time was running out. Still, he had this weird feeling that he was missing something.

Note: The traditional accountant will be ill at ease to notice that Daniel's analysis does not include a single figure for *Standard Costs* per feature, overhead, paid overtime data, allocations for G&A expenses or any other accounting data describing the historical status of current operations and processes. *Cost Accounting* has the insatiable urge to make all costs variable according to production, when in fact they do not behave as such. Doing without those is just as well and Daniel knew that. Today's flow data is all that matters.

Khenbish is Back!

For the first time Daniel was afraid to be a day late and a penny short in finding a solution to a problem. Before wrapping up the financials with the data at hand, he decided to call Khenbish Outis.

Khenbish was a brilliant woman, who's ancestor came from the Altai mountains between southern Siberia and Mongolia. She was an old acquaintance of Daniel's, and a consultant who embraced Dr. Goldratt's thinking a long time ago when each went in opposite directions as Daniel took the path to SAFe. She had always helped Daniel in finding better, logical ways of thinking. Daniel had always been surprised by her suggestions, and was convinced she had godly powers.

He showed Khenbish the financials. Khenbish did not take long to read the business case, the financials and with the observations made in the environment quickly reacted to:

1. The absence of *Constraints Management* and *Throughput* analytics.
2. The expertise mix between in house and consultants.
3. The *Throughput Accounting* report.

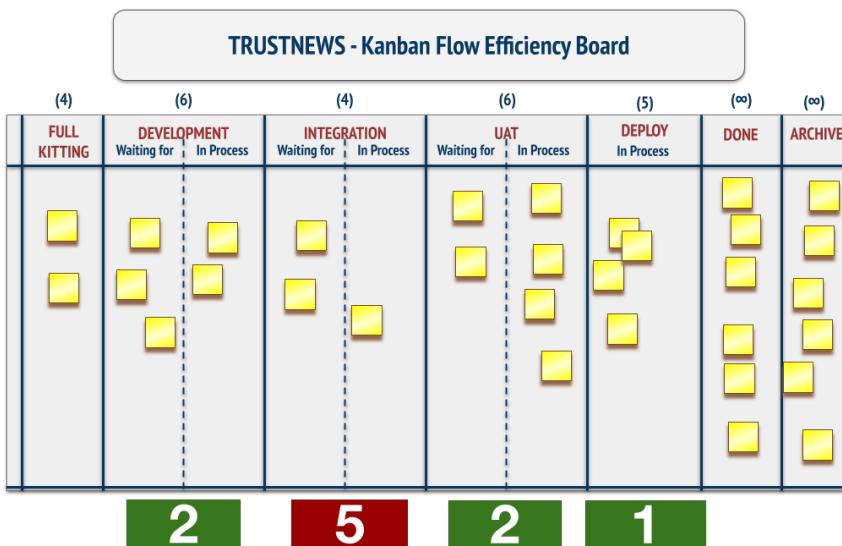
Absence of *Constraints Management*

The first topic puzzled Daniel. He knew about *Constraints Management* but how could Khenbish pick up on the fact that he did not have a *Constraint* under control!

He had to ask Khenbish.

Khenbish told him that the *Tameflow Flow Efficiency* board designs posted on the wall were a significant improvement from the traditional *Kanban Boards* she had seen Daniel operate through the years. She liked the ease with which it could spike *Flow Efficiency* by triggering the right visual stressors and properly account for *Wait Time* while not *In-Process*. While *In-Process*, accounting properly for *Wait Time* with conceptual integrity is done by having waiting columns in front of touch columns as opposed to the numerous “Ready for” parking space buffers that are in fashion these days with traditional *Kanban Boards*!

Khenbish took good care to add at the bottom of each process the average column *Flow Times* from Engineering to show Daniel where the *Constraint* of the system was: INTEGRATION!

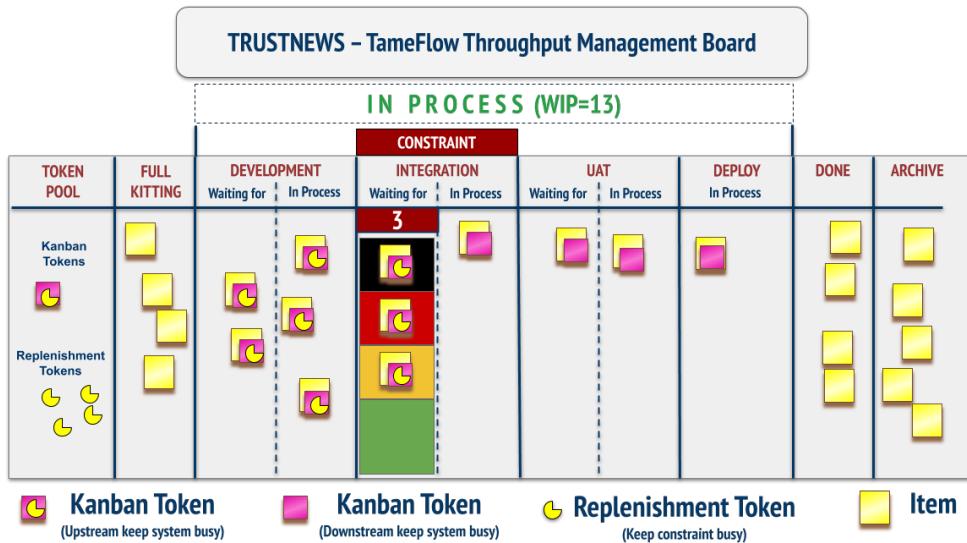


Daniel was surprised and had this internal dialogue about how Khenbish could zoom into the constraint just by looking at *Flow Time* metrics, and without considering how many people/resources were actually employed at that stage? He would later understand Khenbish's perspicacity when the *5FS* will be applied to manage the *Constraint*!

Daniel liked to refer to the entire elapsed time of 10 as duration in a *Cost of Delay* (CoD) context and the basis for calculating his financials. This is how CoD worked in SAFe and he was very proud to use the *Flow Time* as duration for the denominator of his calculation!

Khenbish sighed when she heard Daniel come up with such metrics. Sighed internally.

Khenbish quickly sketched the same *Flow Efficiency* board to increase *Throughput* by making the *Constraint* visible and introducing *Kanban Tokens* as explained later.



Daniel was so enthused that he invited the entire engineering ‘fleet’ to see the board as soon as he saw the *Throughput Management* board drawn by Khenbish.

Daniel shouted : “Folks we have a CONWIP Kanban Board with a defined process attached to it. In KMM, CONWIP is a level 2, low maturity practice and is never linked to a detailed process...”

Khenbish rolled her eyes and immediately suggested to the team to actively read through the [TameFlow Blog](#) for tomorrow.

Khenbish explained that the *Kanban Board* was in fact more than just a CONWIP device. It was, to be precise, a *Throughput Management* board to enable *Flow* through the system. Khenbish pointed out where the *Drum*, *Buffer* and *Rope* components were located on the illustration just above.



Drum: The drum signals the beat of the *Constraint*. The *Drum* beat is represented by the movement of the *Replenishment Kanban* tokens in front of the *Constraint*. The purpose of the *Drum* is to ensure timely *Release* of work into the *Work Flow* as to assure that the work arrives to the *Constraint* just in time for its processing there.



Buffer: The *Buffer* column is meant to hold just the right amount of work in front of the *Constraint* so that it does not starve.

Note: The *Buffer* is also part of the instrumentation employed to produce *Buffer Signals*, leading indicators of risk materialisation. Those signals will be discussed at length in *Chapter 14 - Flow Efficiency*.

DBR and TameFlow Kanban Boards and in Chapter 16 - Introduction to Execution Management Signals.



Rope: The *Rope* is effectively the *WIP Limit*, the capacity put onto the constrained system.

As time permitted it, Khenbish also explained the *Kanban Tokens* that made the circulation of work within the flow system: There are two kinds of tokens on a *Throughput Management* board:



Kanban Tokens: Upstream of the *Constraint*, the purpose of *Kanban Tokens* is to keep the system busy. They are attached to an item together with a *Replenishment Token* (see below). Downstream of the *Constraint*, their purpose is still to keep the system busy, but without the *Replenishment Token* which has been returned to the left of the board to signal that the *Constraint* is hungry for work. Once an item is done and out of the system, the *Kanban token* is detached from the item and returned to the entry pool to the left of the board, and associated with a *Replenishment Token* again.



Replenishment Tokens: Always upstream of the *Constraint*, the purpose of *Replenishment Tokens* is to keep the *Constraint* busy. They provide the drum beat to signal pull of work: a new item may be released into the *Work Flow* only if there is a free *Kanban Token* and a free *Replenishment Token* that can be attached to it. Note well: a *Replenishment Token* alone is not sufficient to allow for a new *Work Item* to be released into the system. Once consumed by the *Constraint*, the *Replenishment Token* returns to their entry pool on the *Throughput Management* board, waiting for *Kanban Tokens* to come back to and be associated with them.

These *Kanban Tokens* act as *Flow* enablers. They are all shown at the bottom of the illustration just above.

Note: The physical distribution of *Kanban Tokens* and *Replenishment Tokens* on a *Kanban Board* provides four additional leading indicators of risk materialisation upstream or downstream of the *Constraint*. The discussion on these leading indicators is out of scope for this book but can fully be appreciated in Chapter 18 of the *Hyper-Productive Knowledge Work Performance* book.

Daniel could not avoid noticing and ironically commenting on how the *Replenishment Tokens* looked like a *Pac-Man*. Khenbish quipped back that, yes! that was done on purpose: to represent that the *Constraint* is like a *Pac-Man*. In order to progress, it must continuously *eat* new work. And that the *Pac-Man* Token is used precisely for that purpose - to signal that the *Constraint* is hungry for work, and needs to be fed - the *Work Flow* needs to be replenished.

In the TRUSTNEWS software engineering value stream illustrated above, we have a *CONWIP* limit of 13 tokens. There are always as many *Kanban Tokens* as there are *Replenishment Tokens*.



CONWIP: For “*CONstant WIP*” - is a pull system similar to *Kanban* but with a limited total number of *Work Items* circulating across the entire system, rather than limited by work stage through *Column WIP Limits*.

Note: *Tameflow Kanban* is an enhanced CONWIP system because there are two kinds of tokens and a *Buffer* (see later) that operate to realize TOC’s *Replenishment Policies* which both minimize *Work in Process* and maximize *Throughput* with respect to the *Capacity* of the *Constraint*.

Now back to Khenbish and Daniel.

After hours, and when Khenbish saw Daniel’s enthusiasm to changing all of the boards he had on the executive floor, she gave a stern warning : “*Daniel all of your Flow Efficiency boards are fine, but find the few that need to be under Constraints Management! Kanban Flow Efficiency boards are appropriate most of the time.*”

Expertise Mix between in house and consultants

After another full day of work helping Daniel, Khenbish noticed that the mix of in-house expertise vs external consultants had not changed since Daniel made the bold move of totally inverting the internal/external mix from 30/70 to 70/30 a while ago when *Flow Efficiency* was low and workers’ skill levels were inconsequential to performance.

| Skill Mix | Flow Efficiency 3% | Flow Efficiency 25% | Daniel's Project Mix |
|-----------|--------------------|---------------------|----------------------|
| Internal | 30 | 70 | 70 |
| External | 70 | 30 | 30 |

The mix is still optimal today in Daniel’s opinion as his staff has been able to improve and get to high *Flow Efficiency* without any outside help.

Khenbish pondered and deemed it appropriate now to suggest to Daniel a slight skill allocation adjustment while still respecting the overall skill mix currently in place.

Here is in essence the dialogue as the team remembers it years later:

“*Daniel. You are wondering why your Flow Time data is experiencing turbulence? Well this is the reason: you are going too fast through the value stream. You have reached the point where you have done all you could with taming Wait Time.*”

“*Khenbish, I don’t understand. Please tell me more as this sounds really interesting*” replied Daniel.

"I hear you," said Khenbish. "Now listen closely Daniel, it's really much simpler than it appears. Remember Goldratt's lessons about Inherent Simplicity? In your current situation, it's hard to figure out the impact that the current workforce mix has on Capacity. If Capacity is not managed, you can expect your metrics to behave in total randomness no matter what you think the cause might be. Now, Daniel we first need to find where the Constraint is and use it as an operational lever. Can you find it?"

"Absolutely," Daniel replied. "I know where it is. It is located at INTEGRATION. I did not know until you pulled the metrics out. But it all makes sense to me now. I have also read about Capacity since I now understand that controlling WIP is not where I am at any longer. I feel energized with this new knowledge and looking forward way beyond next year!"

Daniel followed up with his thinking, "First, we need Productive Capacity. And we need a sufficient quantity at the INTEGRATION Constraint. Then, we need to provide sufficient Protective capacity so that the system does not fall apart. This lack of concern on Protective capacity is often the reason performance metrics go sour everywhere. But not to worry. This can be alleviated once the Constraint is under scrutiny, by managing the Capacity of our system with this three-pronged approach of managing first Productive Capacity, followed by Protective and Excess Capacity. We will get to this as the project unfolds. When Productive and Protective Capacity are not finely tuned, the margin for error decreases."

"I can now add assuredly that when you ignore what I have just illustrated, you are inducing the moving Constraints whack-a-mole style into the system. This is a classical flaw of Lean trying to 'balance' Work Load over all resources. Since the Constraint is no longer in one place, the accumulation of small delays becomes even more nefarious and all these delays just accumulate throughout the system; and they will affect the Constraint too. A time unit lost on the Constraint is a time unit lost by the whole system, forever. Searching for remedies by thinking solely in terms of Flow Time, WIP, Flow Efficiency and Throughput will not help without mastery of Constraint Management and Capacity Management."

Note: The concept of Capacity is discussed at length in Chapter 17 - *Introduction to Full-Kitting*.

"But Khenbish," said Daniel, "I have a question: how could we practically alleviate this Capacity issue?" "This is a bit of a tricky game, Daniel," warned Khenbish. "What specific action could we conceive to specifically shape Capacity? Can we think of something practical that could be done without resorting to WIP as a first response?"

*"Pretty easy, I can find the answer by myself now, Khenbish. I would say that I need to think like Dr. Goldratt and use this new Mental Model you taught me and apply the Five Focusing Steps. The first step is to **Identify** the constraint. It is located at INTEGRATION; the process with the highest Flow Time. The second step is to **Exploit** the constraint. Making sure that it runs uninterrupted at its full capacity. A buffer right at the entry point of the INTEGRATION process is the solution. **Subordinate** to the constraint comes third. Upstream overproduction must be prevented. With this in mind, the Throughput Kanban Board with its token system and Drum Buffer Rope scheduling will keep all actors of the system well informed as to when they should start new work during the daily operations. Fourth, I need to **Elevate** the capacity at the constraint by assigning my top performers at INTEGRATION! And finally, as the closing step, I must **Repeat** this cycle to prevent inertia! This way, I will be able to impact Capacity at INTEGRATION, better protect the Constraint and ride a new learning curve! Wow, I can fix this today, Khenbish," volunteered Daniel "I will assign my best software engineer on a permanent basis right at this point in the value stream and keep my resource mix intact!"*

Khenbish applauded such managerial skills.

The Throughput Accounting Report

In retrospect, Daniel and Khenbish then looked at the metrics that were tallied up before Khenbish's arrival. Option A was bringing 65 more major features per year based on an entire Process *Flow Time* of 9 days and they both knew that these metrics needed adjustments.

First Khenbish had to compliment Daniel on the approach he had taken.

"Daniel, I agree with your thinking entirely in the way you approached your operational analysis," replied Khenbish. *"You did not get caught up with stale cost accounting data ending up wasting your valuable time, which is the right Mental Model to adopt in Throughput Accounting. Congratulations! But that being said, Duration (Flow Time) is a metric that is used in a CoD context. Not in Throughput Accounting."*

"What would you do differently today?" Khenbish asked Daniel

"It is now becoming evident that speed through the value stream cannot be ignored at the operational level where I perform," Daniel admitted. *"But I have the feeling that sometimes a solution with a greater Flow Time could also be beneficial. What do you have to say on that topic?"*

"Right you are Daniel. Right you are! Looking at an increase or decrease in overall Flow Time is not the right answer in itself, it all depends on if it impacts the Constraint in a positive way! But do not take your eyes off the Constraint!"

Daniel modified the spreadsheet immediately to get the right information with flow data at the *Constraint*.

| | Current | Option A | Option B |
|--------------------|---------|----------|----------|
| Total Flow Time | 10 | 9 | 11 |
| Flow at Constraint | | | |
| Development | 2 | 1 | 3 |
| Integration | 5 | 6 | 4 |
| UAT | 2 | 1 | 3 |
| Deploy | 1 | 1 | 1 |

"I see clearly now. The Constraint being INTEGRATION, Option B gives us better Throughput! Not Option A, which has admittedly a better Flow Time altogether."

"Let me now do the proper Throughput Accounting and use the right model and plug in the optimal Flow Time metric at the Constraint."

Operational Throughput (TP) - Features Accounting Report - Option A and Option B based on their respective Constraint

| | CURRENT | Option A | Option B |
|---|---------|----------|----------|
| Capacity - Hours per year at Constraint | 5,840 | 5,840 | 5,840 |
| Flow Time at constraint | 5 | 6 | 4 |
| Operational Throughput (TP) per year | 1,168 | 973 | 1,460 |
| Variation in Operational Throughput | NA | -195 | 292 |

“The incremental benefit is 292 additional features a year for Option B and it must be the one selected over Option A.” Daniel explained

Khenbish continued, *“So Option B makes sense. Does it not now, Daniel?”*

“Now, coming back to the negative differential error of 195 features a year for Option A, it was caused by a faulty analysis done over the shortest entire Flow Time of the Process (9 days) instead of the shortest Flow Time at the Constraint.”

“Khenbish, I have to thank you again. I have a much better perspective on building a healthier and more productive working environment with Dr. Goldratt’s practical management approach.”

Note: Daniel’s story unfolds further in the following chapter!

Takeaways

An honest discussion on the utility of maintaining detailed time sheets is warranted as the value of the data generated by such systems is a relic of the past and a reflection of the need for precise data that *Cost Accounting* requires for reporting purposes.

Let people work instead of maintaining those systems!

In conventional project settings, such historical timesheet archives have no predictive value as technology radically changes along with skills over short spans of time.

Let’s see why this madness must be stopped.

Cost Accounting uses timesheets to increase the value of assets (unfinished projects) on the *Balance Sheet* to reflect capital value.

The longer our death march - we all know what we are talking about here - and the more efforts consumed, the shinier the capital asset. This voids any financial and managerial motivation to finish, kill or drive a project to fruition even when confronted with the sad reality that intellectual work decays with the passage of time.

Talk about accounting fiction. Talk about accounting friction for those who have Dr. Goldratt’s knowledge, which we now have!

We are not saying that proper budget, asset capitalisation and cost control be ignored. (For instance, try to account for cost with a few lines in our project plans, enabling cost control and putting an end to senseless data entry.)

Now, back to improvement models like ADKAR, CMMI or KMM. There is nothing wrong with using the approach of our choice. Just be aware of the following: companies that fail to identify and manage their

true *Constraint* will fall victim to increased overhead charges; explaining why these charges fall out of control everywhere. The case is easy to grasp and goes along these lines: each and every time that we act upon an “improvement” that does not help the *Constraint*, we are financing and maintaining useless tools by seeking local resource efficiency that will increase costs. We are in fact digging our own grave. Relying on *Cost Accounting* data is also a poor choice in making improvement decisions.

Expressed another way - and in no uncertain terms - the use of *Cost Accounting* for performance management and cost control, without managing our *Constraint*, is the main cause of overhead creation and increased operational costs!

“Dr. Lisa” Lang says...

In “Tame your Work Flow” Steve and Daniel explain some of the traps of Cost Accounting and some of the benefits of the alternative, better solution – Throughput Accounting. This basic understanding will serve project managers well and “Tame your Work Flow” should be on every project managers reading list.

“Dr Lisa” Lang, President Science of Business, TOCICO Certified Expert, TOCICO Lifetime Achievement Award Winner, TOCICO Board of Directors, author *Maximizing Profitability, Achieving a Viable Vision, Mafia Offers, Velocity Scheduling System, and Increasing Cash Velocity*.

Clarke Ching says...

Eli Goldratt would have been delighted with this book. Steve and Daniel have elegantly extended his work into the world of knowledge-work.

It's a must read.

Clarke Ching, Author of Amazon best seller “*The Bottleneck Rules*” and “*Rolling Rocks Downhill*.”

8—Show Me the Money

Note : This is the continuation of Daniel's story from the previous chapter.

The next day Daniel presented the business case to Gina, the CEO. She was very impressed by Daniel's acuity and his permanent obsession in bringing forward simple solutions. Yet Philip, the CFO, was disappointed: *"We are not a feature factory here. Why can you not show me how all those features impact the bottom line, and at what cost?"*

"And my differential analysis shows that Option A is better, if you consider how costs are affected." continued Philip.

Unfortunately for Daniel, Gina agreed and requested him to come back with a more economically grounded case.

Daniel knew he was right - after all *Operational Throughput* with Option B would increase significantly, and it stood to reason that it just must have a positive impact on the bottom line. But he was at a loss. He had to call Khenbish again.

Khenbish laughed out loud when she heard what had happened. *"Of course Daniel, Operational Throughput doesn't pay your salary. It is the customer valued outcome that matters. Some customers must value the outcome of your work more than their own money... Now, you must translate all this into Financial Throughput so that Option B cannot be challenged."*

Daniel retorted: *"But how can I do that? We are not producing widgets to which we can allocate costs and profits! We are a SaaS company. The revenue comes from different subscription plans! All of which are supported in one way or another by all of our features. We have advanced DevOps operations with continuous deployment; and we can deploy any single feature as soon as it is ready. How can we account for its contribution to the bottom line?"*

It was time to re-conceive the accounting systems to provide irrefutable evidence of the actual *Financial Throughput*. The current situation was totally inadequate upstream to support matching of revenues to outcomes. For that they decided to get in touch with Karl, the VP of Sales and Marketing and meet him at the coffee shop.

Karl triggered the exchange: *"Hi Khenbish and Daniel, I heard about the need to get the cash inflows more visible and hook them to revenues. I have been there before and it is impossible to do. So I came up with this idea using empirical evidence. We could simply get the needed data from the SCAM (Special Cost Accounting Module) in two easy steps. First, we take the total annual revenues and simply divide it by number of releases and we have the revenue per release. Of course it is an approximation, because some releases come to fruition in the middle of the year. But it is the simplest thing we can do. Then, we get the cost per release and we are done. No change in process, no change anywhere and we have all the data already available. Sometimes I think I am just brilliant! What do you guys think?"*

Daniel's entire body language changed as he looked at Karl dubiously.

"I won't comment on your brilliance, Karl - but what do you mean with 'releases!?'"

To which Karl answered nonchalantly.

"This is the way we run things. We always have. Always!"

Daniel knew that his continuous deployment gave the business an edge but could not figure out why Karl's folks had not realized the value of flow all the way from order to cash. This was not the way it was supposed to be and there had to be something missing somewhere.

It was at this point that Karl nuanced his language: *"Daniel, I know where you are coming from with continuous deployment and we love it here on the business side! Taking care of bugs, fixes and technical debt in that fashion really pleases the business, and it is made available in no time. It is fantastic. Rest assured that we are thankful for you being at the cutting edge of technical thought leadership with all the DevOps stuff you have done."*

Khenbish saw material for positive friction and before it got any further, stated the obvious:

"Daniel, Karl. I smell a consulting opportunity here!" she said jokingly. *"The business needs releases for managing sales and marketing, but what about performance and accounting? This has to come into play and be integrated both with continuous deployment, as well as with sales and marketing. No?"*

Daniel and Karl look at Khenbish in a "So now what?" way without a word being said.

Khenbish continued: *"Let us take a different angle. Karl thinks horizontally with the release notion. There is a Target Scope far down the Value Stream, because that 'release' is the unit of marketing and sales activities. You don't put up a trade show for a single feature! Daniel, on the other hand, has a burn up chart with continuous deployment of functionally complete features in mind. He likes stuff to get done as soon as possible, so that he can remain responsive to the ever changing business requirements - which typically come from your side of the court, Karl."*

Daniel started flashing darts from his eyes, but before he could say anything Khenbish offered the definition of a MOVE and of what was coming up next in her mind to reconcile the two worlds.



MOVE: a *Minimal Outcome Value Effort*: it is the minimal effort that the organization can undertake in order to achieve a desirable outcome and/or produce a given value.

(MOVEs are presented in detail in *Chapter 15 - Outcomes, Values and Efforts in PEST Environments*.)

Both Karl and Khenbish noticed Daniel's body language, which was giving him away again.

Daniel replied quickly. *"Hey Karl, Khenbish. Listen. This is a bad idea. I am agile too and I do not like the idea of having BDUF stuff going on and slowing down my pipeline intake."*



Big Design Up Front (BDUF): is a software development approach in which the program's design is to be completed and perfected before that program's implementation is started. It is often associated with the *Waterfall* model of software development.

Khenbish was about to interject but Karl knew better and reassured Daniel.

"Daniel," said Karl in a reassuring tone. *"Look at the benefits downstream. We don't need to work up detailed designs, but just have a clear definition of what we want to give to our paying customers, and - if I understand Khenbish correctly - establish its value. This is brilliant because we can start planning our marketing activities much sooner and have them better coordinated with your efforts, rather than second guessing what features we will get and how to fit them into some customer valued package of sorts. We can really focus on what the customer truly needs, and avoid those very common cases where features cannot be packaged and sold, and just add cruft to the offering, basically wasting your engineering efforts. What do you say?"*

"Daniel, what is your take on MOVEs?" inquired Khenbish.

"For now and from where I stand, I am in a holding pattern." replied Daniel.

"To conclude guys," said Khenbish, "MOVEs are mainly a coordination construct. They will ease some pain. For the business, it is simply business as usual! And for you, Daniel, it will give you a way to better focus your effort in support of the outcomes and the value that the business needs to provide to the market. Are we all good?"

Everybody nodded to that effect.

The *Financial Throughput Report*

Karl had a business case to make. He told both Daniel and Khenbish that nothing could be better than to tackle a real business opportunity right now - that would at least engage Philip's attention for some minutes.

"Alright!" said Khenbish, "*show us how you go about your business, will you Karl?*" while she was looking at Daniel.

"*Things are pretty straightforward.*" Karl argued. "*Our SaaS offering has three levels : Bronze, Silver and Gold, and the pricing scheme stands at 50K a year for Bronze, 75K for Silver and 90K for Gold.*"

"In the current highest priority business case that I have, there is a strong need for Election Analytics during election coverage. As you know there will be a major election coming up within 2 years, and we want to offer a service that can provide advanced projections and sentiment analysis. It is not only about the revenue, but also the opportunity that we are at the leading edge. Of course the election date is a hard deadline."

Daniel interjected: "*Of course this Election Analytics thing would be a 'Release' from your perspective, Karl. Right?*"

Karl nodded and continued: "*Yes, and in the new terminology it would be a MOVE. Now the offering would qualify for our Silver and Gold clients. We usually do our financial impact analysis as follows:*"

- **New Subscribers (N):** New subscribers are the ones that would be attracted by a new functionality in the release.
- **Saved Subscribers (S):** Saved subscribers are the ones who would end service unless a new desired functionality is delivered with the release.
- **Life Time Revenues (LTR):** The total lifetime revenues over the period that a subscriber uses a release.
- **Release Sales (R):** For a given release, the cumulative sales are equal to the sum of new and saved subscribers with their lifetime revenue *relative* to the release.

$$R = (N + S) \times LTR$$

"Anyway, here is the SCAM data I gathered for getting the green light." concluded Karl. Daniel thanked Karl for being always prepared and told him the news:

"Karl, thank you for thinking so quick on your feet! This is a more challenging request as Gina now wants relevant Financial Throughput data from now on."

Upon hearing the word *Throughput*, Karl almost choked on the coffee sip he was taking and shouted: "*Throughput? What is that!?*" inquired Karl with curiosity.

"Well, we need to make the business case for every step forward we make from now on, Karl, linking all the components to the bottom line at the get-go," said Daniel.

“I see nothing wrong with that and thanks for the heads up by the way.” exclaimed Karl as he continued on looking at Khenbish while saying:

“So you had another move up your sleeve!”

“You can say that again,” Khenbish jokingly replied as she could not wait to discuss the process design of connecting features, modules and releases to MOVEs to Bronze, Silver and Gold revenues in light of TOC and TA thinking.

That being said, she let Karl forge ahead.

“Our three categories - Bronze, Silver and Gold - are clearly the MOVEs that Khenbish speaks about!” said Karl. He took the service offering printout, which he always has in his attache case, folded open the page with the service matrix, took his beloved and precious Mont Blanc fountain pen and encircled his understanding of what MOVEs were. Schematically, it looked like this:

| | Bronze | Silver | Gold |
|-------------------------|------------------|------------------|------------------|
| Base Tier Functionality | A B C D | A B C D | A B C D |
| Mid Tier Functionality | | E F G | E F G |
| Top Tier Functionality | | | H I |

Daniel observed that Karl was contradicting his earlier statement that the Election Analytics “release” should be considered as a MOVE, but Karl insisted that now - considering the complete service offering - it was logically better to reason in terms of subscription classes.

Daniel objected that some features, or even sets of features, were available to all while others only to some of the subscribers, and asked: *“To which MOVE should the basic features belong to then?”*

Daniel and Karl couldn’t seem to agree on what was a MOVE and what went into it.

Khenbish intervened, and asked: *“Karl, has any customer paid you for a single feature alone?”* Karl explained that that was not the case, because it was typically a bundled combination of features that provided the utility value sought for by the customers.

For instance, the set of all basic features was what the Bronze subscribers paid for.

“But wait!” interjected Daniel. *“Don’t both the Silver and Gold subscribers pay for those features as well?”* - of course, they do, confirmed Karl, as part of the higher prices for the higher levels of service they pay for.

Daniel got the insight: *“So we have a set of features, that are used by one or more subscriber categories, and which are paid for accordingly. Isn’t that set of features what actually brings home the bacon? And that should be the MOVE.”*

He grabbed the printout from Karl, and pulled his Mont Blanc fountain pen out of his hand (Karl was speechless), and encircled his view of what a MOVE could be *“They are basically the horizontal partitions in our service offering table! So if your Election Analytics ‘Release’ is offered to both Silver and Gold customers, it is actually one unit of value for them, and one unit of work for my team.”*

Schematically, it looked like this:

| | Bronze | Silver | Gold |
|-------------------------|------------------|------------------|------------------|
| Base Tier Functionality | A B C D | A B C D | A B C D |
| Mid Tier Functionality | | E F G | F |
| Top Tier Functionality | | | H I |

Khenbish nodded.

Karl nodded as well saying: "Yes, Daniel, you're obviously right. That's how we should think about a MOVE. It has to provide value to our customers, and be manageable by us!"

"And because we have continuous delivery," continued Daniel, "the actual partial deployment of features can be decoupled from the business timelines or cycles. In fact we do this all the time, and use feature flags to activate a feature when we are asked to do it (actually it would be a whole set of features), even though that might happen much later than when it was materially deployed. And we can do this in full confidence that nothing will break upon activation because we use Behavioural Driven Development from the outset."

Karl commented: "Clearly Daniel, we ask you to deploy or activate - as you say - those features only when the whole marketing/sales machinery is ready to get them out the door. We call it the 'Roll-out.' And it is at that point that our SCAM module starts counting the revenue streams. But I see the brilliance. The MOVE would allow you to organize features in any way you need from a technical perspective, with module and feature flags, and - what did you say? - the driver tests?... All the while we would have a unit of value and accountancy, and you would have a clear target to cook out of the feature soup as the Master Chef that you are!"

Daniel didn't like the sweet talk, but he agreed. Then he protested that some features were used by multiple customer categories, and immediately started doing proportions and allocations.

Khenbish stepped in and observed: "Daniel, you must avoid double counting and make sure TH and OE are taken into consideration only once. This is similar in nature to a resource having Excess Capacity that you account one time only for TH and OE purposes. Naturally, you will have to recognize that the TH of a MOVE, given your multi-level subscription plans, can come from different categories of customers after it has been deployed. Remember that this is Financial Throughput. Furthermore, it is additive. You can simply add them up per project until you reach your full capacity - a strategic construct that Cost Accounting, ROI, COD, NPV just cannot support. Also the technical timelines can be decoupled from the business timelines. The advantage of this lies in calculating the Financial Throughput, because we can refer to the natural lifecycle of Karl's releases from a business perspective."

Daniel was excited. "I see, there is beauty in this inherent simplicity of Dr. Goldratt's work. So rather than trying to find a way about how to fit one feature in multiple MOVES, we just need to be smart about how we define a MOVE! Any feature belongs to one and only one MOVE. We don't need to do any allocation exercises. The MOVE can easily represent functionality valued by different categories of customers, who might pay different prices for the same functionality because they are on different plans, but at the end what one pays

can be added to what another pays. And that means that, from an accounting perspective, a feature belongs to one MOVE, and we only deal with the value produced by the MOVE! It makes it even simpler, because a MOVE is thus a horizontal partition of our subscription plans' feature matrix - as I said before!"

Karl broke in with even more commotion: *"And Daniel, because you release these features as we need them - with your feature sets and feature flags - that keeps our notion of 'Release' intact, and we can continue to structure and do our familiar customer lifetime revenue calculations as we have always done. I see that we have found a way to bridge your best practices with our sales and marketing practices. I wonder why we have never done this before? But how do we tie this back to what Gina and Philip are waiting to see?"*

For Daniel - still overwhelmed by the excitement - it was yet another new *Mental Model* to develop.

The clock was ticking and it was time to reconnect with Gina and Philip to show them the new approach to revenue recognition and costing. They would prepare a standard *Throughput Financial Report* for the Election Analytics module. Daniel has estimated salaries for the development costs at \$350,000 with internal employees.

Karl informed his peers that Gina and Philip had approved the merits of that business case but they needed to be reassured. While Gina really wanted the Election Analytics module because it would be a major news event at the next trade show, Philip had his reasons to be negative - very negative, actually. So showing that *Constraints Management* had an impact on their top priority would certainly leave a permanent mark.

It was already mid afternoon and all three decided to go for a bite at the pub, as they had skipped lunch. Needless to say it turned out to be a working lunch.

Khenbish and Karl had already ordered, as Daniel was slow to show up. Upon entering the pub and before sitting down, Daniel expressed that he was at a loss to figure this thing out. Karl looked in total agreement. Khenbish felt the need to appease both of them.

"It is not easy doing that kind of analysis the first time. Trust me, I have been down that road before!"

"So what would either of you do as a first step." was the question directed at both Karl and Daniel.

Karl started: *"We need the Releases sales (R)! And for that it is pretty obvious that we need to assign values to: New subscribers (N), Saved subscribers (S) and Lifetime revenues (LTR) - over the period that a subscriber uses a release."*

"Fantastic," said Khenbish. "Let's head back to the office and write this down!"

"No need for that," said Karl as he gave the basic numbers he knew inside out, and summarized them sketching a table on a blank page in his notebook. It looked like this:

| Election Analytics | |
|-------------------------------|-----------|
| Number of Subscribers (N) | 100 |
| Saved Subscribers (S) | 50 |
| Lifetime Revenue (LTR) | \$3,975 |
| Release Sales (R) = (N+S)*LTR | \$596,250 |

Daniel then put things in perspective and looked at Khenbish, *"How do we get to the magical Financial Throughout reports, Khenbish!"*

"OK. Let's cut to the chase," Khenbish replied, "It has been a long day and tomorrow morning, we are facing Philip again. We need a Financial Throughput report for the option selected,"

"In Throughput Accounting we look at the differential between options. In this case we need to have the data and the baseline of the current situation. Now, Karl and Daniel, you quickly need to produce the differential

Net Profit and Return on Investment figures for each of Options A and B. Once you have that, we're done."

To Khenbish's surprise, Karl and Daniel were able to sketch down the report on the fly as if performing an improv. It looked as follows:

| Election Analytics | |
|----------------------------------|-----------|
| Number of Subscribers (N) | 100 |
| Saved Subscribers (S) | 50 |
| Lifetime Revenue (LTR) | \$3,975 |
| Release Sales (R) = (N+S)*LTR | \$596,250 |
| Number of Features (F) | 96 |
| Variation in Investment (I) | \$30,000 |
| Monthly Operating Expenses (MOE) | \$50,000 |

| | Option A | Option B |
|--|-----------|-----------|
| Capacity of Election Team (C) | 16 | 24 |
| Estimated Duration in Months (M) = F / C | 6 | 4 |
| Variation in Operating Expenses (OE) = M * MOE | \$300,000 | \$200,000 |
| Variation in Throughput (TH) = R | \$596,250 | \$596,250 |
| Variation in Net Profit (NP) = TH - OE | \$296,250 | \$396,250 |
| Return on Investment (ROI) = NP / I | 988% | 1321% |

Since the project had already been up for evaluation by Gina and Philip, Daniel was already familiar with it because he had to provide his estimates for the project's duration. He reckoned that it would need 96 new features (**F**). Karl had indicated the investment (**I**) of \$30,000 that had already been spent in consulting services to refine the market research; and the additional monthly operating expense (**MOE**) of \$50,000 for the services of a Big Data and AI specialist for the duration of the project.

Then Daniel took over, while both Karl and Khenbish watched. He showed the two options, and how they differed in terms of delivery capacity (**C**). The capacity of the election software engineering team was 20% of total capacity, but depending on how they engaged the *Constraint* with either Option A or Option B, they could deliver 16 or 24 features a month respectively.

From this Daniel derived the expected software development duration in months (**M**): 6 months for Option A and 4 months for B.

Then he quickly calculated the variation in *Operating Expenses (OE)*, *Financial Throughput (TH)* and *Net Profit (NP)*; and also *Return on Investment (ROI)*.

"Wow," shouted Khenbish "You are one step ahead of me here!" In the same breath, Khenbish concluded: "We now have everything we need with Net Profit and ROI values!" affirmed Khenbish.

"You mean we are ready for Gina and Philip tomorrow?" Karl and Daniel said together.

"Yep," said Khenbish contributing one last bit of information, "We are good to go! And again you were right the first time Daniel - the Operational Throughput report seldom lies!"

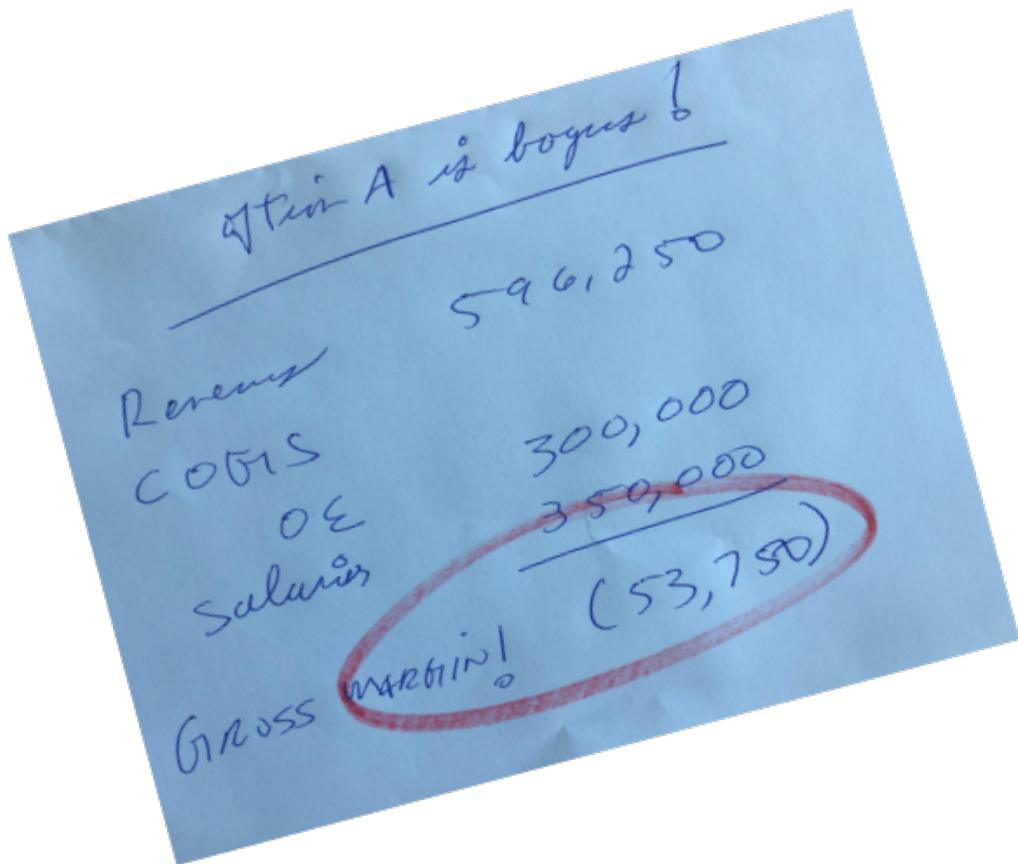
Karl had a few twists of his own to slip into the discussion and concluded, "This report speaks volumes and is not that hard to generate. It will again broaden the downstream view that I have of DevOps. This is exactly what they mean by opening the path to double loop learning and I am starting to like it."

The “GOAL” Line

The next morning Daniel and Khenbish presented their findings.

Gina, Philip, Khenbish and Daniel were all present and sitting down at the center table in Gina's office. Daniel distributed his findings. Philip knew the business case - and he knew it was a loser! He was just wondering how Daniel got around it while not using the SCAM data, since he noticed that he had not logged in lately - except for entering his timesheets.

Philip looked at the report quickly and said it was easy to grasp. But then he turned blue; jumped off his chair; folded his report in half; turned it face down; stole Khenbishi's Kaweco pen; and grabbed the red marker on Gina's board. He wrote down a few lines and put this piece of paper in the middle of the table:



"This accounting is so flawed, even for Option A, that it would get me fired anywhere. It shows a Net Profit of \$296,250 on your statement - when in fact it has a loss of at least (\$53,750)!" fumed Philip! But Daniel interrupted and meant to ask what was 'COGS':

"It is Cost of Goods Sold," answered Philip. "And it must include salaries before the gross margin. I can understand that you have omitted variable G&A and selling expenses which must also be taken into account, but not to subtract salaries from revenues to get a gross margin is something that I can't accept as a CPA."

Gina was interested in hearing Philip forging ahead, *"We need to cover all of our costs in the long run and for that, at least cover our variable costs in the short run! This is why having a positive gross margin makes sense all the time!"*

Gina looked at both Khenbish and Daniel to see who would pick up the ball. Khenbish and Daniel looked at each other.

A few seconds later, Daniel spoke up.

"Philip, the \$350,000 salaries of full time employees dedicated to the development of the solution does not help in decision-making. No matter which option we were to select, they would still have to be paid. And if we were not doing this project, we would still have the same monetary obligation. Do you agree?"

Philip nodded with his head going sideways and said to Gina, *"He has a point but it is not how we run a business. We must cover all of our costs. It is what the SCAM is there for!"*

"Philip, the SCAM has a very solid regulatory purpose and it is needed. It goes into all the minute systems that we have and gets all the data we can imagine. It is great and I admire what it does for its user, especially for external stakeholders and consumers of key information."

"That is so kind of you, Daniel! Thank you!" said Philip.

"Now, to each his own. What I need is decision-making support. Today's news. Relevant data to make money. You serve me with precise data that is too delayed in time for me to act upon. For example, the cost per feature provided by your system is for production levels that are no longer realistic. This makes no sense to me anymore. Now that I understand Throughput Accounting, you and I can make managerial decisions with a snap of our fingers with no need for historical data. For instance, the head office overhead costs are not impacted no matter what my production level is, and yet I get charged for those plus G&A, and other fixed costs."

Gina signaled to Daniel that they all got the message and to sit down with the rest of them. The tension in the room was at hazardous levels: finance and tech were on a frontal collision trajectory, and knowing Philip and Daniel, it would not be easy to find a compromise.

Gina glanced at Khenbish, who stood up and said: *"We know that we are all committed to the success of this company. That each and every one does their best. And as managers, this means making decisions. But these decisions stem from our understanding of the world - from our Mental Models. And at this moment - ladies and gentlemen - we are witnessing the clash of two Mental Models that leads to conflicting decisions. Now is not the moment to be right or wrong. Now we have to figure out what is best for the business."*

Khenbish paused for a moment to let those words sink in.

She looked at Philip and said: *"Philip, if you didn't know about accounting practices, what would happen if we actually did what Daniel is suggesting? Would there be an extra \$396,250 on the bottom line or not? Do you see anything that could materially prevent that from happening?"*

Philip: *"Yes, I can see how that could happen, but..."*

Khenbish abruptly interrupted Philip and loudly proclaimed: *"Nobody will blame anybody for not doing what they don't know." She quickly scanned the room, focusing an instant on everybody's eyes - to make sure the message got through.*

Then she continued: *"You all know the Story of Herbie, that you have heard me tell so many times, don't you?"*

"Keep it DRY, Khenbish!" interjected Daniel, who had heard the story one time too many (the "DRY" was referring to the software engineering maxim of Don't Repeat Yourself).

"No worries, Daniel. I won't repeat it again! But you recall how the boy scouts team helped Herbie by offloading his backpack so they all could proceed more quickly towards the base camp. Right?"

Khenbish paused again; and the silence confirmed that they were all thinking about that episode.

"At times what is overloading our own backpack are our Mental Models. We need to get rid of those that don't help us, and pick up better ones. That is how we can improve the way we think, and how we can learn to make better decisions. And remember there is no dishonor in being Herbie. Everyone will help you so we all can reach our common Goal."

She walked around the table and put her hand on Philip's shoulder. *"Philip, as the CFO of this company, it might be embarrassing for you to have ignored a \$396,250 opportunity. Nobody will blame you for not knowing what you don't know. You made the best decision you could with the Mental Models you had, to the best of your abilities and experience."*

"But now that you saw the way of reasoning that Daniel and Karl came up with, and you understand that there is nothing that would objectively prevent those \$396,250 from materializing, isn't Throughput Accounting worthwhile to consider?"

Philip bowed his head, nodding in acknowledgement.

Khenbish asked: *"Philip, the next time there is a high stake business decision, instead of just looking at your spreadsheet in isolation, will it not be better to work together with Daniel and Karl and see how the whole company system can achieve its Goal, helping each other, and in particular helping out the Herbie of the moment? And who knows, maybe there will be many more \$396,250 to discover that you would otherwise ignore."*

Philip stood up, took Khenbish's hand and shook it: *"Thank you Khenbish. I see now. It is humbling. I thought I was leading the troupe, but in reality I was the Herbie of the moment. Thank you, Daniel and Karl, for offloading my backpack. Yes, let's get to the basecamp together before the sun sets! From now on we will never again miss any such \$396,250 opportunity just because I don't think with my own head!"*

Philip further commented that it was fascinating to grasp how a better understanding of the *Constraint* and the time dimension could lead to so much better performance; and how relying on *Cost Accounting* would make that \$396,250 in *Net Profit* disappear!

What could be more convincing that *Constraints Management* was worth its salt!?

Daniel was amazed by Philip's turn around, and for the first time he had really witnessed what Khenbish was always calling the *Enlightened Self-Interest*. And the magical ingredient was *Constraints Management* that caused each and every one's self-interest to be in the one same place: *"The Goal."* They just needed to be enlightened to see it that way. Through a *Mental Model!*

In fact, Gina was jubilant to be enlightened herself and to discover how all this new way of thinking had brought together engineering, sales and marketing, finance and... herself, for once all agreeing with no objections on what was the best way forward.

Maybe the *Unity of Purpose* that Khenbish mentioned so many times was possible after all. And Philip and Daniel seemed to have found new ways to trust one another, setting a new foundation for building a *Community of Trust* into the organization.

She needed no more evidence that *Constraints Management* indeed delivered what it promised. And not only for the bottom line results!

Throughput Accounting - Pain Relief for the C-Levels

What can we learn from Daniel's story?

We saw in our last few chapters that *Throughput Accounting* brings a new vocabulary that acts as a bridge between the accounting views of the CxO level (Gina and Philip) and the operations and sales views (Daniel and Karl).

C-Levels experience high degrees of frustration. They feel like the rest of the organization does not understand them, and is not even willing to listen to them. There is a disconnect between the *Mental Models* of the C-Levels and those that are ruling the lines.

Conversely, the people doing the work don't have the "financial literacy" to understand the *Mental Models* of C-Levels, and often think that they are out of their minds, because they experience on a daily basis that there are conflicting priorities between the desires of the C-Levels and what is "common sense" for them!

This level of frustration exists. It is there because there is no common language, no shared *Mental Models* that can help the C-Levels really see what matters, and help all to see and communicate coherently about the same thing.

Throughput Accounting offers the medium to bridge this disconnect. It is based on the *Mental Model* of managing the *Constraint*; and when this is done positively, benefits will be both operational as well as financial.

Throughput Accounting was the rational solution that Daniel introduced into the business context only after he understood the impact of the *Constraint* in his operations. Eventually, both Gina and Philip rallied when the impact of that same *Constraint* could be connected to the bottom line, in a logical and conceptually consistent way.

The same reasoning was supporting operations, sales and finance. The added value, then, was truly the humane resolution, where trust and purpose could transcend functional status and rank.

With *Throughput Accounting*, both Gina and Philip will not feel challenged as being the rational ones, the 'keepers' of the number and the ones dealing with cold hard facts! *Throughput Accounting* gives them a different, more relevant view, about which facts are truly important and had ignored until that point. Their *Mental Models* simply did not contemplate the presence of a *Constraint*, let alone the huge impact it has on all aspects of performance.

With *Throughput Accounting* the C-Levels can keep on counting their numbers - as always - and now they will not only count the right numbers (the *Throughput* numbers) but also be counted as trusted members of an extended team, that speaks the same language, and shares the same *Goal*.

Throughput Accounting will alleviate the friction that always endures between the C-Levels and line operations. The organizational impact of *Throughput Accounting* can be viewed as a motivational hygiene factor. When present, it is not noticed and business runs smoothly with no friction between the participants. When it is not, its absence creates tension, confusion and even open conflict as to what is the best course of action - sadly, in most places, this is the norm, because no better alternatives are known.

With the discovery of *Throughput Accounting*, we now know that there is a better alternative. It is an *obvious* alternative in hindsight that is not *self-evident* from the outset.

It is a better alternative that leads to a *Community of Trust* with a *Unity of Purpose*.

Takeaways

Throughput Accounting is its own virtue:

- By focusing on increasing *Financial Throughput* first, on the basis of TP - *Operational Throughput*, morale goes up as there are theoretically no limits to what can be done with positive energy. As a second step, TA then moves on to reduce *Inventory* (WIP) followed by *Operating Expenses* (OE). On the other hand, *Cost Accounting* works in the opposite direction and focuses on cost-cutting - causing immense human suffering as you cannot cut costs below zero!
- By treating *Inventory* as a liability, and not an asset, thereby reducing the monies required to finance the excessive WIP.

- By permitting unanimity-based decision-making once the *Constraint* is visible and known to all, in lieu of repetitive and painful consensus building that continuously drain knowledge-workers' sanity.

Note : A distinction with a difference is warranted at this point. *Flow Metrics* are nowhere to be found in our accounting books or systems. But they are accounting topics; both at the *Operational Flow* and *Cash Flow* levels! Also, adopting the visual indicators that come with *Flow Metrics* will ease the burden of communication and the transaction and coordination costs associated with such activities.

One final word on DevOps. The market, the tools, the vocabulary are growing at record speed. Agile has always been a promoter of automation and it has a direct link with agility. So in the spirit of Agile, let's "maximize the quantity of work not done" by understanding *Capacity* and *Constraints Management* - even more so in a DevOps perspective.

Read this chapter again. It is not only about the numbers. It is also about improving culture and decision-making, with science that works. It is about data-driven focus.

Savio Neville Spiteri says...

What frustrates me most in my working experience so far is seeing so much wasted energy and unnecessary conflict by very well meaning, hard working, dedicated individuals - execs, managers and "workers", all trying their best to do what seems best for their organisation. Everyone would be pulling in different directions, although everyone seems to have the same purpose.

What really strikes me in TameFlow is how through the use of shared "Mental Models", true organisational alignment is achieved - what the authors call "Unity of Purpose" - and organisations are able to learn how to focus on what really matters and achieve meaningful and lasting improvements through systematic and methodical approaches.

This book presents a mix of theory and practical advice to help you understand the (sometimes counter intuitive) concepts and start applying them, whatever your context.

Make no mistake - this is not an easy read, but one well worth your attention if you really intend to succeed in any change initiative you are about to embark on.

Savio Neville Spiteri, Agile Coach

PART 4—Maximizing Business Value in Knowledge-Work

When we are dealing with...

- multiple **Projects** or **Products**,
- multiple **Events** (or deadlines),
- multiple **Stakeholders** and
- multiple **Teams**

... we are confronted with **PEST**. We need to gain control!

Conventional approaches, both Agile and others including the popular *Kanban Method*, simply do not offer us the tools to manage such turbulent environments.

On the other hand - as we will learn in this section - the *TameFlow Approach* thrives in and is perfectly equipped to deal with PEST and VUCA environments.

Not only will we develop ways to be in charge of such environments, but we will also learn how to do so and maximize the business value that is being produced.

Stefan Willuda says...

The TameFlow Approach is something worth to wrap one's head around. The book shows easily and clearly that constraints management can and should be applied to knowledge-work and how to pursue it step by step.

The authors challenge well-established mental models on how organizations may be 'managed' and offers alternatives approaches that enable higher performance and customer satisfaction without strain or stress.

This is not mainstream management literature, it's new thinking for a complex world.

Stefan Willuda, Lead Agile Coach at Idealo Internet GmbH.

9—Constraints in the Work Flow and in the Work Process

In the earlier chapters we have seen several instances where conventional thinking was a less than optimal solution.

In reality, conventional improvement approaches often appear to look better than what they really are.

Let us understand why.

In *Chapter 5 - Where to Focus Improvement Efforts*, the proposal put forth by David, notwithstanding apparently delivering sooner and costing less, actually had almost no impact at all; and yet it incurred a permanent ongoing cost to finance the improvement initiative. Most practitioners will object that this sort of worthless improvement effort is not seen in reality; and if observed, the underlying approach would be ditched in no time.

The missing observation is that most improvement methods strive to *improve everything, everywhere all of the time*.

So, in the scenario of Chapter 5 where David's proposal was aimed at step B, a more realistic description would be that any "systemic" improvement effort would affect all steps (B, H and G). Therefore, by *also* affecting the *Constraint* (step H), these improvement initiatives *do* produce positive outcomes. The overwhelming positive outcome (on H) will typically mask out the negligible or even negative contributions on the *non-Constraints* (B and G) - but it will all appear as if that kind of "systemic" improvement brings home results. It does, but with waste in terms of initial *Investment* and especially additional long term recurring *Operating Expenses*.

Often - in virtue of the positive results observed - the method and approach will be applied again. Yet it might not yield the same kind of positive effect because in the meantime the *Constraint* - the target - has moved somewhere else.

These methods are shooting blindfolded in the dark for an ever moving target. Not knowing what to shoot for, they blast at everything, everywhere, all the time. Naturally, they might get lucky and hit the target that really matters among all the others that do not. When we miss, layers of costs are not too picky. They will just grow, accumulate and take our bottom line for a long ride downhill.

By focusing on the *Constraint*, the *TameFlow Approach* is much more selective. It is more like using a rifle equipped with high precision aiming devices.

What difference does this make in practice, one might ask?

To start, it avoids the waste of time, effort and money on striving to improve on things that do not matter. But, even more significantly, as we saw in the traffic jam example of *Chapter 6 - Introduction to Throughput Accounting*, when a *Constraint* is clearly identified, it affords us the luxury to exercise *unanimity-based decision-making*. We do not need to waste brain cycles and have endless discussions to build consensus about what to do next as is always the case in Agile ceremonies and meetings. This boosts the speed and quality of decision-making, across the entire organization.

High performing teams and organizations have a natural tendency to do this sort of decision-making intuitively, primarily because of working together for a long time and because of a shared common experience. Such teams will *look at reality* with a *common understanding*. Yet, this is an expensive proposition. Most teams and organizations will not have had the time to ride the shared learning curve required to develop that common understanding and to have lived through such shared experiences. Being able to find the *Constraint* is like a short-cut; an enzyme; a catalyst. It allows the entire organization

to have an immediate shared understanding of where to focus thought and action. It short-circuits long decision-making processes, as both the people on the ground as well as top managers will have the same view of reality. It provides a sense for metrics that can be used to drive the business rather than to keep workers accountable toward achieving irrelevant targets (“KPIs”) and maybe even chastise them if they fail.

Furthermore, as we have learned in the previous chapters, with a focus on *Operational Throughput* on the one side and on *Financial Throughput* on the other side, having sophisticated *Constraints Management* practices in place will bring systems-wide operations and financial management aligned on the same tuning fork.

In ordinary situations, where there is no *Constraints Management* and no *Throughput Accounting*, companies are being set up for failure by creating sterile conflicts which just waste efforts in infights rather than focusing on customer needs and market penetration.

VUCA and PEST

Given the above, it is obvious how knowing where the *Constraint* is remains of paramount importance. If we are not able to identify the *Constraint*, the ideas of the *TameFlow Approach* are inconsequential.

Yet, even a renowned expert like David J. Anderson (one of the creators of the *Kanban Method*) who is certainly very much aware of the *Theory of Constraints*, has given up the quest of trying to understand where *Constraints* are in a knowledge-work setting. Hence the development of the *Kanban Method*, that takes a shotgun approach, and more recently, with the *Kanban Maturity Model* that resembles the deployment of an army of shotguns. It offers nothing in terms of relieving knowledge-workers from the overburdening of chasing improvements at all costs, at all times, in all places.

Incidentally, if you're attacking everything, you don't have to worry about the *Constraint* moving. Therefore if it is moving somewhere else is an irrelevant case - and you might just as well claim that the *Constraint* cannot be identified in knowledge-work because it is constantly moving.

In a VUCA (Volatility, Uncertainty, Complexity and Ambiguity) world, one can easily jump to the conclusion that there is no *Constraint*, because everything changes so rapidly, all the time. Apparently, there is no stable base from which a *Constraint* can be identified through empirical observation or derived through analytical deduction. It all seems very random: the “*moving Constraint syndrome*” becomes so relevant that *Constraints Management* is deemed impossible.

Yet, as Dr. Goldratt taught us, the more complex the situation appears to be, the less degrees of freedom will govern it. And there should be a way to determine where the *Constraint* is.

It is evident that in any situation where there is teamwork or concerted organizational effort happening, there exists some sort of stability. This is where the metaphor of *The Jeep, the Jungle and the Journey* comes handy.



The Jeep, the Jungle and the Journey: The way “*we do things around here*” (the *Work Process*) with the team, the organization is like a *Jeep*. We are in control of it. We set the direction and drive. We fix and maintain it to the best of our capability. The business domain (the *Work Flow*) wherein the team or organization operates is the *Jungle*. It is full of dangerous beasts, traps, swamps. The real challenge comes about when we actually set out to move through the *Jungle* - this is the *Journey* (the *Work Execution*) which will present even more unknowns and surprises.

Note well: we can make sure that the *Jeep* performs in the best conditions so as to be able to adapt quickly and react to whatever surprises appear on its path.

The functioning of the *Jeep* is our *Work Process*, while the path we are uncovering in the *Jungle* is the *Work Flow*. Our *Journey* moving along that path is the *Work Execution*.

The important insight here is that the real *Constraint* of a system comes from the combination of our *Work Process* and the (maybe surprisingly random) shape and form of *Work Load* that hits us through the *Work Flow*.

To keep the *Jeep* working at its best, we must know what is the limiting factor constraining its performance.

If we use a team *Kanban Board* and collect *Flow Time* metrics, we can quickly arrive at the conclusion that the *Constraint* is the column that has the longest average in-state *Flow Time*. This image works well as long as we are in the scenario of linear *Work Flows*.

As soon as we start having *Work Flows* that handle multiple *Projects* or *Products*, multiple *Events* (or deadlines), multiple *Stakeholders* and multiple *Teams* - what we call a “**PEST**” environment - this sort of linear vision diverges ever more from reality. We will have to manage value streams or even value networks, wherein streams can split, converge, combine and affect one another in all possible ways.

The moving *Constraint* syndrome will become a continuous nightmare with no apparent resolution. The moving *Constraint* syndrome has nothing to do with the observation of a moving bottleneck on a *Kanban Board*, yet it is so easy to confuse the latter for the former and reach all sorts of distorted conclusions or detrimental decisions. Again, at the risk of sounding repetitive, and when in doubt, a *Constraint* is under targeted, active and deliberate management, while bottlenecks are not.

It is here that we must start to consider the “*shape and form*” (statistical distribution) of the *Work Load* hitting the system as a real *Jungle*. As we will see that shape and form is absolutely critical in determining where we will have the *Constraint* in the overall system. If we have multiple teams, then the *Constraint* could be any one of those teams in the *Work Flow*. Once we have identified *that team* in the *Jungle*, then this is where we need to look at the process of *that team* (i.e. that team’s *Jeep*), and see where *that team* has its *Constraint* in their *Work Process*.

The way we actually perform - the *Work Execution* - while traversing the *Jungle*, also contains surprises. This is the *Journey*! In later chapters we will learn how to quickly detect that the *Constraint* is moving from one team to another; and react accordingly.

This is how we keep readiness for the VUCA world. We realize that the *Constraint* can and will move depending on the shape and form of the *Work Load* hitting the system. Then, depending on that shape and form we identify the *Constraint* team. Within that *Constraint* team we focus on the *Constraint* in its *Work Process*. Yet we are prepared to see it move to another team, due to unforeseen circumstances (Mr. Murphy) that happen during the *Work Execution*.

But, to begin with, how do we know where the *Constraint* is in the *Work Flow*, especially in an environment infested with PEST!?

The answer is straightforward: we need to monitor mathematically (and of course visually too), how work is piling up *in front of the individual teams* and then figure out *which of those teams has the longest queue of work* waiting for it.

It is easy to state, but much harder to actually do in a PEST environment. Even more so if you are using conventional *Kanban Boards* with *Column WIP Limits* that distort the shape of the natural queues in the *Work Flow*.

Before delving into how to survive in a PEST environment, let’s consider a much simpler scenario that will allow us to become familiar with and start reasoning about the *Constraint* in the *Work Flow* and the *Constraint* in the *Work Process*. (We leave the more advanced treatment of the *Constraint* in the *Work Execution* for later chapters.)

Constraint in the Work Flow and Constraint in the Work Process

We need to understand in more detail the distinction between the *Constraint* in the *Work Flow* and the *Constraint* in the *Work Process*.



Constraint in the Work Flow: is the *Work Process* which is part of the *Work Flow* and that is facing the greatest *Work Load*.

Note: As we will see later, we will think of the *Work Process* as a *Kanban Board* representing a single team, while the *Work Flow* is the stream of work that spans any configuration of multiple teams. The *Constraint in the Work Flow* is the team that is the *Constraint* of all such teams.



Constraint in the Work Process: is the *Process Step* that presents the longest average *Flow Time* in that *Work Process*.

In particular we need to appreciate that the *Constraint in the Work Process* lives *inside* the *Constraint in the Work Flow*. The ultimate *Constraint* is always a *Constraint in the Work Process*, but in a PEST environment, where there might be multiple concurrent processes happening simultaneously, the specific constraining *Work Process* will change in a dynamic manner, depending on the *Work Load* distribution that is coming down with its own shape and form impacting the entire *Work Flow*.

Note: As mentioned, in later chapters we will also learn about the *Constraint in the Work Execution*, which is created by the dynamically evolving circumstances. It can be temporary, or it can evolve and become a *Constraint in the Work Process*. To effectively manage the *Constraint* we need to be familiar with all three notions, and that the *Constraint* is determined by the *Work Flow*, the *Work Process* or the *Work Execution*.

Let's first understand what this difference is, with a simple model that is still linear, but that highlights how the two elements interact. As we will see later, the notion of the *Constraint in the Work Flow* and how it interacts with the *Constraint in the Work Process* becomes even more relevant when we have a PEST situation. Yet this simpler viewpoint allows us to develop an understanding of the concept we need to master. Again, we are trying to grasp reality through a simplified *Mental Model*.

Where is the *Constraint*?

One of the practices used in *TameFlow Kanban* is that of identifying the *Constraint in the Work Process*, by looking for the work state that takes the longest average *Flow Time*.

In traditional *Kanban*, bottlenecks are typically identified visually by looking for queues and/or starvation *on the board of any single team*. The work state in front of the one that is being starved could be a bottleneck (or not). A work state where there are queues could also be a bottleneck (or not).

Naturally, when work state *Column WIP Limits* are employed, the real queues are more difficult to see (because they could be induced by the very fact that *Column WIP Limits* are employed), but - at least - starvation is always visually evident.

Practitioners of *Kanban* often refer to the *Five Focusing Steps* of the *Theory of Constraints*, to handle such bottlenecks. Yet, such an application might not be warranted, because those are more likely just bottlenecks and not the *Constraint*.

The difference between the two concepts might be subtle, but it is fundamental to understand in order to know how we can manage them. Let's look deeper at these different perspectives.

Note: The *Theory of Constraints* also makes similar distinction, though does not use the terminology of *Constraint in the Work Flow*, *Constraint in the Work Process* or *Constraint in the Work Execution* (which will be described in *Chapter 19 - Execution Management in PEST Environments*). For instance by using *DBR Scheduling* at the portfolio level, TOC is managing the *Constraint in the Work Flow*. Likewise by using *Critical Chain Project Management*, TOC handles the *Constraint in the Work Process* and in the *Work Execution*. It must be understood that ultimately the *Constraint* is one and only one at any given moment in time - like TOC teaches. This categorization is not about different *types* of *Constraints*, but rather about how to think about them in relation to knowledge-work where (in the *Work Flow* or in the *Work Process*) and when (during the *Work Execution*) require us to adapt in different manner and with specific remedies in order to safeguard our *Financial Throughput*, or more generally our *Goal*.

The “Painting Gadgets” Example

To introduce the concept, we refer to a fictitious example where we have to paint a number of items, and then those items are hung for drying. The drying process takes 10 hours. There might be more or less painters that do the painting job.

In January 2014, in an email exchange about this topic, Mike Burrows proposed this example for discussion (paraphrased here):

One painter can paint one item per hour. Two can paint two per hour and so on. Assuming plenty of spare rack space for drying, the 10 hours required for drying (supervised by one person) has no effect whatsoever on throughput; items will leave at the same rate they arrive. The longer activity is not a throughput Constraint; if we want to get more throughput we add painters, not drying supervisors.

Naturally, it is an astute example. It is good to start with such a physical example, because then TOC concepts are actually relevant and more easily applied (TOC has more than 30 years success in manufacturing, so it is *extremely* good at handling these physical situations).

To make this example realistic, let us now assume that we do not have “*plenty of spare rack space*” and that we certainly know how much space we have, and that is one potential bottleneck.

There is a clear distinction between “*bottleneck*” and “*Constraint*”:

- A bottleneck is simply a resource that has more demand placed on it than its capacity to deliver.
- A *Constraint* is the bottleneck with the *least* capacity in the entire system - relative to demand - and it can be effectively considered as the “*Constraint*” only once we decide to manage it.

Constraints are always under managerial control. Bottlenecks are not - and should not be - unless they become the *Constraint*.

Let us first think about the *Work Flow* because it is what is most visible; and what the *Kanban Method* typically deals with. If the overall capacity of all painters available is less than what can go through the drying space, the reasoning is sound. We can consider “*painting*” as the work state that is the *Constraint*. Applying the standard *Mental Model* that says that to get more capacity we must hire more painters to address the situation is a natural reflex. This is exactly what is originally suggested by *Kanban* too, and it makes sense.

Note: We stress that we are building the argument here by applying the *standard* way of thinking - and not by strictly applying the 5FS as TOC would suggest. We do this in order to highlight where to look for the *Constraint*, because its location changes dynamically depending on the context. We are not focusing on putting TOC into practice by playing by the TOC book (we will do that in later chapters). For the moment we are concerned about learning *where* to look for the *Constraint* in order to more easily identify it and subsequently manage it.

At a certain point we add so many painters that the “*painting*” work state is no longer the “*Constraint*.” The *Constraint* moves to the “*drying*” state. This happens the moment we have added so many painters that they produce more items than can be placed at any one time on the drying racks. The interesting additional observation, from a TOC perspective, is that by elevating the *Constraint* (the painters), at a certain point the *Constraint* will move: in this case from “*painting*” to “*drying*.”

Note: Our own actions are moving the location of the *Constraint* not by happenstance but by deliberate management decisions.

Suppose we have done that. We have effectively operated on the *Work Flow*, and we get to the point where the *Constraint* is the “*drying*” stage. Because of the nature (a chemical process) of the stage, it appears we have hit something that cannot be further optimized. It will need 10 hours of drying, no matter what; and the rack space is limited.

From Work Flow to Work Process

Time to switch gears. Thus far, we have progressed according to the perspective of *Kanban*. Now we take a look at what we can do from a *TameFlow Approach* perspective.

We start by looking at the *Work Process*, instead of the *Work Flow*. We do not consider the painters and the drying supervisors. Instead we look at a single *Work Item* going through the whole process.

(Imagine we are a patient going through a hospital. What do we experience in terms of delays from that perspective?)

The painting process goes through 1 hour of painting and 10 hours of drying. *The 10 hours of drying are the Constraint of the Work Process*, no matter how many painters we might have. It is the single work state that (on average) takes the longest for the whole process of painting one single item.

The only way to improve the process is to reduce those 10 hours. Maybe we could devise the usage of some kind of baking oven so that by applying temperature, the drying time would go down to 3 hours.

Evidently, we would incur the *Investment* of building the oven, and the *Operating Expenses* of keeping it running. *Throughput Accounting* would tell us if it is worth our while. If it is, then we have elevated the *Constraint in the Work Process*. Drying is still the *Constraint* in the process (3 hours drying is more than 1 hour painting).

However, it could be that the elevation is high enough, that the *Constraint in the Work Flow* goes back to the painters. Back to square one. Time to hire more painters, as suggested originally.

But how would we achieve this elevation of the “drying” stage?

In one of two ways: either by improving the *drying process* even more (maybe by getting an even more efficient baking oven that does the drying job in 30 minutes rather than in 3 hours); or by maybe getting more baking ovens (just like we were induced to hire more painters at the beginning) and increase the *capacity* of the drying stage, without changing its actual process.

In the first case, by reducing the drying time to 30 minutes, we would have moved the *Constraint in the Work Process* to the “painting” stage (because the “painting” stage still takes one hour, while now “drying” would be half of that with a new drying process).

In the second case, the *Constraint in the Work Process* will still be with the “drying” stage by adding drying capacity (because “drying” would still take three hours, which is more than the one hour required for “painting”).

In both cases, however, the *Constraint in the Work Flow* moves from “drying” to “painting” simply because in the aggregate, we now can dry more items than we can paint.

Constraint in the Work Process and *Constraint in the Work Flow* are complementary.

They are the two different perspectives, which necessarily need to co-exist, depending on what is truly impeding *Flow* through the system.

In the real world, we would keep on switching focus between one and the other, depending on where the *Constraint* is at the moment.

This is not a heuristic, but is simply conventional *Theory of Constraints* thinking and its effective application. One of the innovations with the *TameFlow Approach* is to add this process focused thinking on top of what we ordinarily would do with *Kanban*. *Flow Time* metrics are the key. Long running *Flow Time* state averages will reveal the *Constraint in the Work Process*.

As in the above example, we might have one *Constraint in the Work Process* (3 hours drying), yet another, different *Constraint in the Work Flow* (depending on the relative capacity of painting vs drying).

Shape and Form of Demand

Suppose we are in the scenario where we have sufficient painters and a good drying facility, and let's suppose that the painters can produce more items per time unit than the drying facility can actually handle (in the same time unit).

The key question now is: where is the *Constraint*?

Without looking at the shape and form (the statistical distribution) of the *Work Load* demand we just cannot truly say. We would like to determine where the *Constraint* is simply through *observation*, even

though we are dealing with non material knowledge-work. We will learn more about this in *Chapter 11 - Finding the Constraint in PEST Environments*.

But we can argue as follows:

We know that the *Constraint* in the *Work Process* is the drying phase, because it takes the longest average time. It is also the *Constraint* in the *Work Flow* because it limits the amount of painted items that can flow through the system. Yet if the demand of the incoming *Work Load* increases to such an extent that it exceeds the capacity of the painters, then there will be a visible queue of work piling up in front of the painters. In this situation the painters become the *Constraint* in the *Work Flow*, despite the fact that the drying facility is still the *Constraint* in the *Work Process*.

Notice the terminology we use here: while we could talk about the “bottleneck” in the *Work Flow*, we are more precise and say it is the “*Constraint*” in the *Work Flow* (all the while the *Constraint* in the *Work Process* remains the same).

In *Chapter 5 - Where to Focus Improvement Efforts* we were very keen to make sure we don't confuse bottlenecks with “the” *Constraint*. With that perspective we now have a situation where both painting and drying are bottlenecks (demand is higher than either can handle), but we consider them *both* as *Constraints*. So why this confusion? Because we are building *two* mental models: one is focusing on how a *Work Load* moves through the *Work Flow* of a system, while the second one is concentrating on how any single piece of work is actually handled in the *Work Process*. The reason for having these two perspectives will be clearer in later chapters, where we will start looking at non-linear *Work Flows*, something that the *Kanban Method* struggles with.

With this simple and linear example, though it might seem that we are splitting hairs, the distinction between the *Constraint* in the *Work Flow* and *Constraint* in the *Work Process* will become more significant as we examine more challenging PEST environments in the next few chapters.

Fluctuations in demand can vary dramatically in a VUCA world and for that, observation of queues is the right remedy.

While the notion of the *Constraint* in the *Work Flow* and the *Constraint* in the *Work Process* are essential to master, keep in mind that any apparent discrepancy between the two notions will find a resolution once we discover the significance of the *Constraint* in the *Work Execution* in later chapters.

Relation to *Special Cause Variation* and *Common Cause Variation*

In the *Hyper-Productive Knowledge Work Performance* book, the topics of *Special Cause Variation* and *Common Cause Variation* are examined in great depth; and they will recur throughout the pages of this book too.



Special Cause Variation: is variation that you can specifically identify. Typically it comes from *outside* of your process.



Common Cause Variation: is a common, recurring variation. Typically it recurs many times, and it resides *inside* your process. It is so common, that if you can identify and reason about it, you might be able to point out a *root cause*.

Notice that *Special Cause Variation* and/or *Common Cause Variation* are just as likely to hit either *Touch Time* or *Wait Time* indiscriminately. However, identifying both bottlenecks as well as the *Constraint* in the

Work Flow will (typically, but not always!) expose *Special Cause Variation*. Identifying the *Constraint* in the *Work Process* will (typically, but not always!) point us toward *Common Cause Variation*.

Being able to identify and act on *Common Cause Variation* is absolutely critical for achieving lasting and significant improvements in the system's performance in knowledge-work according to Donald Reinertsen. Assigning every variation to *Special Cause Variation* is common in knowledge-work. It is also wrong. *Common Cause Variation* is expected in a manufacturing setting, but is ignored as an improvement driver. This oversight has been transposed blindly to knowledge-work. Instead, with the *TameFlow Approach* we seek out to understand the causes of *Common Cause Variation* in order to be able to make systemic and long lasting improvements.

Note: The *Kanban Method*, while recognizing - in theory - that there are different kinds of variations, adopts a qualitative approach that - in practice - assigns everything indiscriminately to *Special Cause Variation* because it is not equipped to detect *Common Cause Variation*. In the *TameFlow Approach*, the instruments of operational execution management (in particular through *Execution Management Signals* combined with *Reason Logs*, *Frequency Analysis* and *Root Cause Analysis*) give us the means to identify and manage both kinds of variation.

Being able to distinguish between the two types of variation becomes essential in order to improve in a knowledge-work setting, because in this context, identifying and acting on *Common Cause Variation* has much higher long term impact in addition to immediate benefits.

Being attentive to where the *Constraint* is materializing - in the *Work Flow* or in the *Work Process* - can be of guidance to what kind of variation to expect (with all the caveats and exceptions, of course), and hence speed up our analysis and decision-making.

Takeaways

The Story of Herbie taught us that the first step is always to identify the *Constraint*. Once the *Constraint* is known, then we can apply the other steps of the *Five Focusing Steps*.

In knowledge-work, the location of the *Constraint* is not obvious. We need to learn how the different dynamic elements of knowledge-work interact and almost collude to make the *Constraint* appear in one place rather than another. Thus we need to prepare our observational abilities by learning to distinguish where a *Constraint* can appear - and this chapter highlights at least two purviews that should always be present: the perspective of the *Work Flow* and the perspective of the *Work Process*. In later chapters we will discover that there is even a third perspective, that of the *Work Execution*.

Remember: “*Step One: Identify the Constraint*” has to be done right!

Mario Latreille says...

I've been looking for principles and concepts to further evolve my clients' agility and throughput of value, without frameworks.

"Tame your Work Flow" provides great insight and ideas for the next level(s). I'll be using the material in this book for several years to come, as I imagine many others will.

Real thought leadership is rare - thanks to Steve and Daniel for providing it.

Mario Latreille, Enterprise agility facilitator, mentor and coach; Fellow, Government Transformation Institute; Founder and Managing Director, Ludu Consulting Partners.

10—Understanding PEST Environments

PEST environments, as introduced in the previous chapter, are those dealing with:

- multiple **Projects or Products**,
- multiple **Events** (or deadlines),
- multiple **Stakeholders** and
- multiple **Teams**.

The example examined in the previous chapter highlighted the difference between the *Constraint* in the *Work Process* and a *Constraint* in the *Work Flow*. That case was still very linear. Now we want to extend the thinking to cover the situation where we have a true *PEST* environment, and in this instance we will refine our understanding about the *Constraint* in the *Work Flow*.

The situation becomes tricky when the *Work Flow* spawns off in multiple streams and then converges again. Any one of the distinct streams can contain the *Constraint* in the *Work Flow* of the entire system. Once we identify that stream, we can focus on finding the *Constraint* in the *Work Process* that is located inside that stream alone.

It is here that the two-dimensional, sequential and linear nature of a *Kanban Board* can distract us from seeing what is really happening. More sophisticated methods of synchronization, coordination and prioritization are needed, especially if we want to truly apply *Constraints Management* and if our objective is to create a reliable high-performing organization.

The examples illustrated in the following pages are taken from the *TameFlow Simulation*, which you can find described in detail in the *TameFlow Games* booklet, available for download for free at this book's [bonus page](#). In fact, these concepts are more easily understood if experienced through a direct hands-on game or simulation.

In particular we want to understand how to:

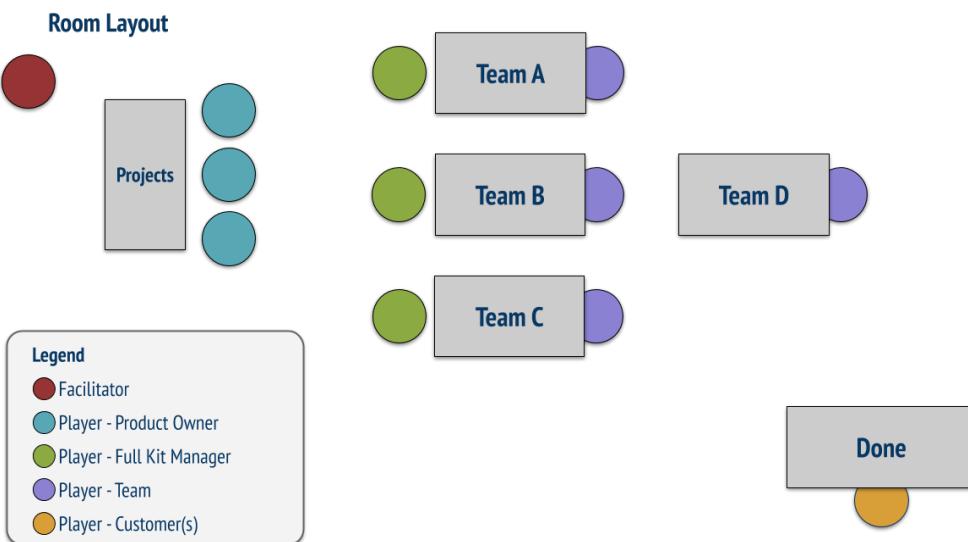
- Identify the *Constraint* in a non-trivial *Work Flow*. (See *Chapter 11 - Finding the Constraint in PEST Environments*.)
- Schedule projects to minimize *Work in Process* and maximize *Throughput*. (See *Chapter 12 - Drum-Buffer-Rope Scheduling*.)
- Schedule projects so as to avoid conflicting priorities between multiple stakeholders. (See *Chapter 13 - Portfolio Prioritization and Selection in PEST Environments* and *Chapter 15 - Outcomes, Values and Efforts in PEST Environments*.)
- Select a subset of projects that maximizes the economic return. (See *Chapter 13 - Portfolio Prioritization and Selection in PEST Environments*.)
- Determine the sequence of execution of the projects that provides the highest economic benefit sooner. (See *Chapter 13 - Portfolio Prioritization and Selection in PEST Environments*.)
- Learn how to balance effort vs. outcomes and business value. (See *Chapter 15 - Outcomes, Values and Efforts in PEST Environments*.)
- Be in control of the execution of all projects. (See *Chapter 19 - Execution Management in PEST Environments*.)
- Have a system that can gather the right information and make it available to the right decision maker at the right time. (See *Chapter 14 - Flow Efficiency, DBR and TameFlow Kanban Boards* and *Chapter 16 - Introduction to Execution Management Signals*.)

- Ensure that *all work is visible* and that *work flows uninterruptedly*. (See *Chapter 17 - Introduction to Full-Kitting*, and *Chapter 18 - Full-Kitting as Ongoing Executive Activity*.)
- Provide top management an unprecedented means to effectively *control and govern complex PEST environment*. (See *Chapter 19 - Execution Management in PEST Environments* and *Chapter 20 - Operational Governance in PEST Environments*).

Overview of the TameFlow Simulation

While the details on how to perform the *TameFlow Simulation* are given in the bonus booklet mentioned above, we will describe in general terms what happens during the simulation, and in particular what the lessons learned are.

The *TameFlow Simulation* is performed with six tables laid out as shown:



There are four **teams** (Team A, Team B, Team C and Team D), three distinct *Product Owners* (representing diverse stakeholders with conflicting priorities), and some further roles that are introduced at later stages in the simulation. Notice that Teams A, B and C are the “frontline” teams, meaning that the actual work starts with them. When a project is released into the *Work Flow* it will be given simultaneously and in parallel to these three frontline teams.

There are also **customers** representing the stakeholders for final approval of the deliveries. Teams can handle any type of projects belonging to their specialty, i.e. the type of coins assigned to them.

A bag of mixed Euro coins in denominations of: **1c, 2c, 5c, 10c, 20c** and **50c** represent the work to be done. Each coin represents a single unit of effort, and its face value represents the business value delivered. The amount and types of such coins is unknown (to those participating in the simulation, as is the case with the amount and type of work coming from our customers until their demand reaches us).

A **project** is represented by a random mix of exactly 10 coins. The simulation will involve ten projects, each one composed of a random mix of 10 coins. In total, during the simulation 100 coins will be moved around by the participants, divided in 10 projects.

To simulate the **work performed**, each coin will either be flipped or passed from one team to the next (according to the *Work Flow* that we will see in a moment). Each coin is “processed” by either the effort of one coin flip (effort = 1) or simply by passing it, i.e. “sliding” it over (effort = 0) to the end of the table. The flipping and passing actions are there just to simulate that some activities take more time than others.

Note: Flipping a coin is a one step manoeuvre. It simply consists in turning it on the other face.

Simulating a PEST Environment: Initial Setup

A project is a (random) combination of 10 coins - taken out of that distribution of 100 coins - which represents the *Work Load* that each team has to perform to execute the project. The *Work Load* will have a “shape and form,” in other words, a statistical distribution. There could be so many coins of one kind, and so many coins of another kind, that the work to be done will be different for each team.

There will be multiple project requests hitting the system. In the simulation only ten projects will be considered, but of course - in the real world - a large company may have to deal with hundreds, or even thousands, of projects at any given time.

These ten projects will each be assigned at random to one of three *Product Owners*. Any project that is approved for execution will have to go into a queue (the “*Backlog*” in Agile parlance) in front of all the teams which make up the system. When a project is pulled from the *Backlog*, it is “released” into the *Work Flow* - in other words, it moves through the system of teams according to how it needs to be processed.

To make it more realistic, the *Product Owners* have their *bonuses* tied to how quickly they can deliver projects. (And, of course, the following year’s budget will be allocated more favourably to the better performing *Product Owners*.)

In an initial scenario, the teams will accept projects on a *first come, first served basis*. The teams will be oblivious to any ordering done by agreement between the *Product Owners*, and will happily serve anyone of them who shows up at their desk with some work to be done.

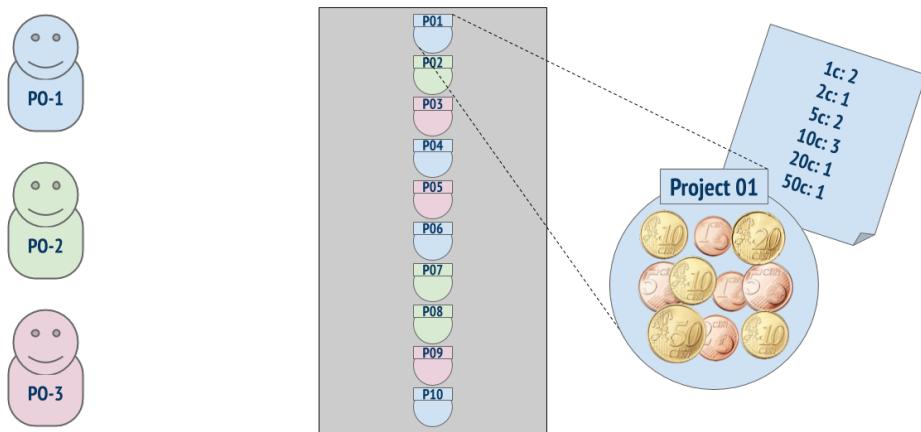
Since *collaboration* is highly valued, the *Product Owners* have to *agree on the overall priority* and sequencing of the projects from the outset, and rearrange the *Backlog* accordingly. Every individual *Product Owner* will order their subset of projects as to maximize their bonuses which are tied not only to on-time performance but also to each project’s value. Each project has a “business value” equivalent to the sum of the face value of its coins. A coin represents one unit of *Work Load* and the “business value” is indicated by the coin’s face value.

If the negotiation goes well, the ordering of the *Backlog* queue represents the ordering of value for the business. Ideally, the project that is at the head of the *Backlog* will be the most valuable one and it will start first, with the others to follow.

When a project is picked up for execution by its *Product Owner*, the *Product Owner* will visit each of the front line teams, and hand out the respective coins that those teams are specialized to handle. Clearly moving from one team to the next just to get the project started is a way to simulate the coordination and synchronization effort intrinsic in PEST environments.

If, on the other hand, the *Product Owners* are not able to quickly negotiate and agree on a sequence of projects, then you can resort to some quick *competition* to establish the ranking. Just like in normal companies, there will be winners and losers. The important thing is that there is a *nominally* agreed upon sequence of execution of the ten projects.

The *Product Owners* will line up their common view of the projects sequenced by value in the *Backlog*. In the following illustration we can see the setup that we have in mind. We have three *Product Owners*, each one owning a subset of the ten projects (notice the color coding used to represent the ownership).



The ten projects are lined up in a *Backlog*, with the highest business value project at the top, and the lowest at the bottom. Each project contains a random selection of ten coins out of the 100 coins that we have in our bag. In the illustration we see an example of what might be the composition of coins of the first project.

Remember that each project is a random mix of ten coins. Each project should also have a **bill of materials** to represent that mix. The bill of materials can be seen as a reflection of *Work Load* (the amount of coins, some of which need to be processed in certain ways, and others in different ways), and as *business value* that needs to be delivered (the sum of the face value of all coins in the project).

For example, in the illustration above, we see that *Project 01* contains:

- two **1c** coins,
- one **2c** coin,
- two **5c** coins,
- three **10c** coins,
- one **20c** coin and
- one **50c** coin.

Together, the 10 coins in *Project 01* carry the “business value” of **114c**.

In the execution of the simulation, one essential function of the *bill of materials* is to keep track of which coins belong to which project. In the figure above we see the contents of coins of the first project only, but when the simulation is running, there will be multiple projects being moved around on the tables simultaneously. The participants will have to figure out how to keep them in order. After all we are simulating a PEST environment.

To have metrics, we will also perform time keeping and record how long each project takes from start to finish.

The Work Flow and Work Process for a Single Project and Four Teams

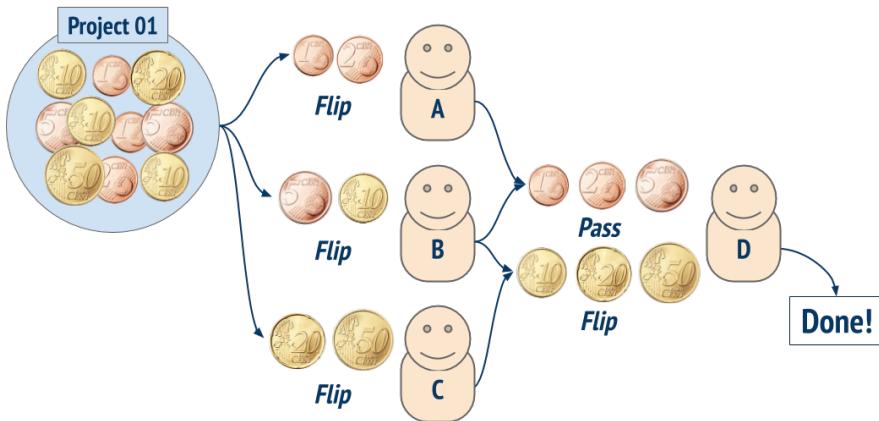
The *Work Flow* is not linear. In fact, the whole purpose of the simulation is to learn about how to handle complex *Work Flows* containing different *Work Processes* (here represented by the different teams).

When a project is started, i.e. it is “released into the *Work Flow*,” the *Work Items* (the “coins”) are distributed to teams A, B and C according to their coin types. All these three teams are front-line teams, which receive the initial work orders.

Once any one of these three teams has performed its work on its part of the project, its partial deliverables are handed over to Team D. Team D will do the integration and assemble together the three partial deliveries into one final product which is handed back to the project’s *Product Owner* (and *Customer*).

The *Work Flow* looks as follows.

Work Process Rules - Which Team works on Which Types of Coins



Round 1 - The Warm Up

For any individual project, the *Product Owner* has to hand out the coins to the three front-line teams (A, B, or C) according to their denominations (and work expertise), as follows:

- All the **1c** and **2c** coins in the project are handed over to Team A.
- All the **5c** and **10c** coins in the project are handed over to Team B.
- All the **20c** and **50c** coins in the project are handed over to Team C.

Note that the *Product Owner* does not give any work directly to Team D. Team D will receive its *Work Load* as three partial deliverables which are handed over from the three frontline teams.

The intent here is to simulate the fact that in complex project environments different teams will handle different types of work.

It is not only the *Work Flow* that changes according to the types of coins, but also the actual *Work Process* is different.

To represent the difference in *Work Processes* we make a distinction according to the types of coins as follows:

- Teams A, B and C will simply flip *all* the coins they receive.
- Team D will flip the “golden” coins only (**10c**, **20c** and **50c**).
- Team D will simply collect any “bronze” coins and assemble them together with the other coins without doing any flipping (**1c**, **2c** and **5c**).

Once teams A, B, and C have flipped all of their coins, they will hand them over to Team D.

Note: When doing this hand over, it is important that the coins of different projects do not get mixed-up once they arrive on Team D's table. Some *housekeeping* activities will be necessary, and the teams have to figure those out also.

All *Work Items* pass through Team D and then reach the finish line. We might imagine that Team D performs some kind of final “integration” activity that brings together the work produced by the other three teams, and then delivers it to the original *Product Owner* (who in turn will bring the final work to the *Customer*).

Team D will ensure that all the coins that are listed on the project's bill of materials are still present and that none were lost in action during the processing of the upstream teams. With just one project, this seems trivial; but with 10 projects moving at the same time, keeping track of which coins belong to which projects will become a challenge!

Team D also has to take care of the bronze coins - it will not perform any active work (i.e. flipping them), but just inspect to see if they are present as listed in the bill of materials, before assembling them together with the golden coins into the final delivery.

Remember that the bronze coins handled by Team D are the only instance where coins are not flipped, i.e. there is no effort expended; and yet those bronze coins must be part of the final delivery (after all, the three front line teams did perform some value adding work on the bronze coins by flipping them).

Team D will put together all coins belonging to one project and prepare them for delivery. The *Product Owner* will collect the finished project and deliver it to the *Customer*. The *Customer* will do a

final inspection and decide whether to accept the project or send it back for *rework*. The time taken for the entire project is finally recorded.

The *Customer's* acceptance criteria are not specified, and left to the participants to define. They could - for example - require that at least half of the coins must be delivered heads up. Any reasonable acceptance criteria will do, as long as most projects will pass most of the time.

If a project needs rework, it is sent back to the bottom end of the *Backlog* queue in front of the *Work Flow* to be reprioritized for reprocessing. Reprocessing means to *run through the whole process again*. Meanwhile the time keeps on ticking for the project.

Just like when adopting *TameFlow* in practice, we start by merely trying to understand what happens in the existing *Work Flow* and *Work Process*. A random project is picked at first, and we run through the *Work Flow / Work Process*. We just want to ensure that every participant knows what to do, and that the *Work Load* actually flows and is processed according to the rules.

Execution time is recorded by the *Product Owner*. Once the rules of the *Work Flow* and *Work Process* are clear, the players should know how things move around, and there should be no hesitation as to how to process the incoming work.

Round 2 - First Estimation

A second round is played with the same original project used in the warm up round. Since the learning ("training on the job") is done, the expectation is that the second round will be executed in less time than the first one.

However, before the second round is run, the players are asked to give an estimate of "*How long will it take?*"

The second round is executed and the estimates will be hit quite reasonably.

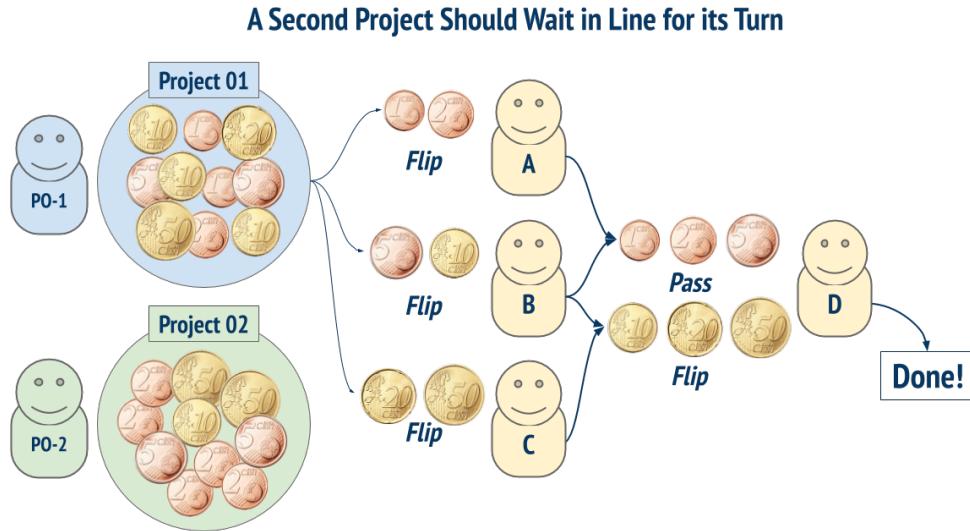
Lesson Learned: When there is only one project, it is possible to provide reasonable estimates even in the presence of non-linear *Work Flows* and non homogeneous *Work Processes*.

Round 3 - Two Projects and Conflicts of Interests Immediately Emerge

The next round will be a small step toward making the situation more realistic. A second random project will be picked by a second *Product Owner*.

This will immediately show how *conflicts of interest* arise within the organization due to the fact that the organization must also service the requirements of a second project - or more precisely a second *Product Owner* as well as the represented stakeholders.

This round can be introduced by announcing that the company is growing and is therefore taking on more work. A second *Product Owner* enters the scene, with a bag of ten random coins. The coins in this second project will have to go through the same *Work Flow* and the same *Work Processes*, with the same rules as the first one.



The second project is placed below the first one in the *Backlog*, meaning it is ranked second. The first project is exactly the same that was executed previously in the warm up round.

The two *Product Owners* are reminded that their projects are very important for the company's business and that they must be delivered *as soon as possible*. Also, they are cautioned that their individual bonuses and next year's budget allocations will depend on how well they fare.

Since Project 01 happens to be the first in line, it will be started first, and it will be timed just as in the previous rounds. Even Project 02, the second project, will be timed in the same manner: timing will begin from the moment the first coin of a project is handed over to one of the front-line teams, and will stop when the entire project is delivered to and accepted by the *Customer* with all 10 original coins.

Note: The reason we say “all 10 *original* coins” will be clear as the simulation unfolds. As we will see, in the upcoming iterations, there is a high risk of coins from different projects being mixed up upon arriving at the table of Team D.

Now the housekeeping chores will necessarily have to become more effective, because neither *Product Owner* would want the coins of their projects to be mixed up with the coins of the competing project. They would both lose! So we let the teams figure out how they can work without mixing up the coins of the two projects, so that both are delivered with integrity, maybe just suggesting that a *bill of materials* could help. Of course, we all value the collaboration that arises!

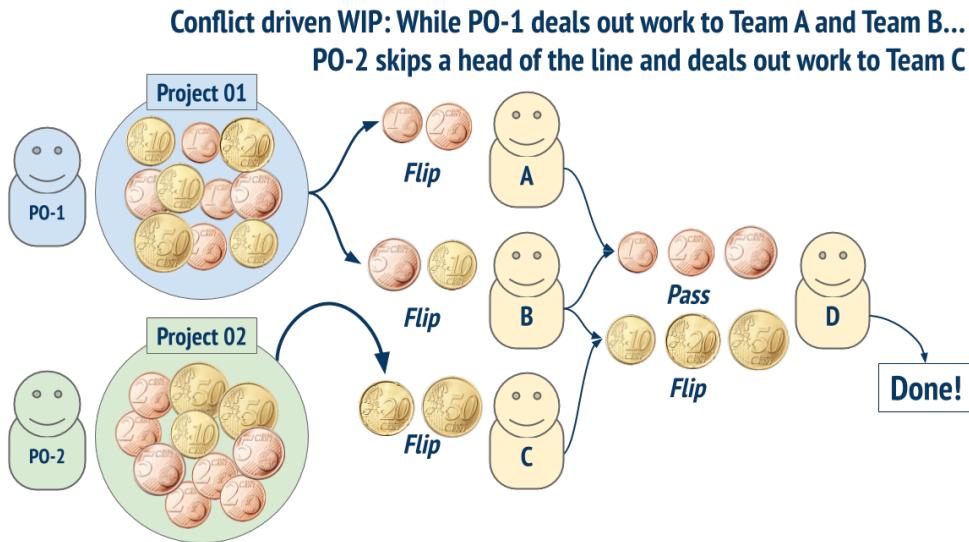
Skipping the Line Becomes the Norm

Since the teams work on a first come, first served basis, the second *Product Owner* can decide to start their second project as soon as there is a front-line team available to handle their coins.

So while the first *Product Owner* might be busy releasing the first subset of coins to Team A and Team B, the second *Product Owner* could “skip line” and release their subset of coins to Team C if that team were idle.

This will be a real possibility, because handing out coins to one of the front line Teams requires some time: the *Product Owner* has to take all the coins of their project to the first team, pick the right kind of coins that can be processed by that Team, and then take the remaining coins and move on to the next team. Clearly while the first *Product Owner* is doing this with the first and second team, the second *Product Owner* will see their opportunity to start their own project with the third team which is just waiting.

In short, because the assumption is that the sooner work is started, the earlier it is delivered, and the *Product Owners* are rewarded for early and on-time delivery, they will have conflicting interests as to who gets to utilize the work *Capacity* sooner - notwithstanding the nominal and collaboratively established face value ranking of the projects!



Again, the simulation round starts, both *Product Owners* are asked to estimate the duration of their projects.

The participants are also asked: “Where do you think the Constraint might be?“

Bonuses, Rewards and Budgets Create Conflicts of Interests

When the third round is executed, it will already be a fact that the estimates will turn out to be much less reliable. The duration and due time will still be estimated reasonably well, but already we will detect a

deterioration, compared to what was witnessed with one project alone.

The competing *Product Owner* (PO-2) will *push* work into the system and increase the *Work in Process*, recreating the very scenario we examined at the beginning of the book when discussing flawed *Mental Models*.

These conflicts of interests not only increase the *Work in Process* but in so doing introduce instability, and make estimation less reliable.

Besides, the effects of skipping to the head of the line and increasing *Work in Process* can also be caused by individual HiPPOS (Highest Paid Person's Opinion) pushing their agenda into the *Work Flows*.

Lessons Learned: Bonus, rewards and budget allocations create conflicts within the organization and there will be internal, unhealthy and noxious competition for the available resources. The cost of coordination (the “house keeping”) increases exponentially with the number of projects; estimates deteriorate; and the climate becomes tense. The *TameFlow Patterns of Unity of Purpose and Community of Trust* don't have a chance to develop. (Furthermore, since in such situations there is no common *Goal*, it is not even possible to apply the TOC conflict resolutions techniques.)

With respect to the question of where the *Constraint* is, with only two random projects, it is very likely that Team D will be identified as the *Constraint*. It is also likely that a queue will build up in front of Team D, while the front line teams will have already finished their work.

This makes sense. Not only is Team D the integration point that occurs after the work of the first team, but it is also performing more work. The first three teams share the burden, work (i.e. flip the coins) in parallel, and hence have approximately one-third of the work each to perform. Team D receives all coins but needs to flip only half of them (the golden coins), and hence has approximately one-half of the work to perform, but sequentially. Most participants will confidently indicate that Team D is the *Constraint* in the *Work Flow*.

But they will be in for a surprise!

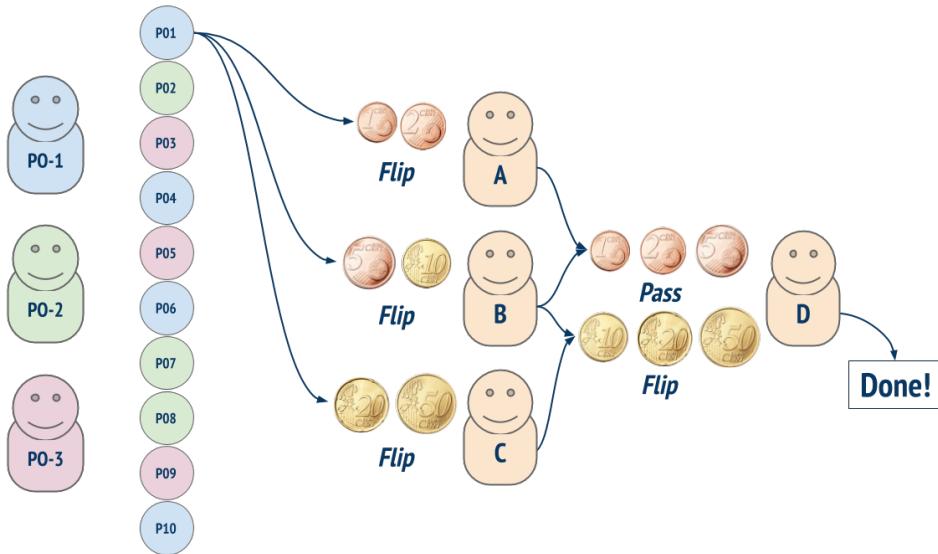
Round 4 - Ten Projects and Three *Product Owners*

In the fourth round, the full impact of multi-project environments will be exposed. A third *Product Owner* is introduced into the simulation. As before, all *Product Owners* must deliver their products ASAP, or their own bonuses and budgets will suffer.

The three *Product Owners* will now be responsible for delivering *ten* projects, each one consisting (as before) of ten random coins (picked from the bag of 100 coins). They will be given a few minutes to work out and establish a priority sequence for the projects. They can “negotiate” (i.e. the loudest or more dominant will probably win the contest); or run a competition or contest - like having coin tossing duels, as there is no shortage of coins!

In any case the projects will be listed in the planned and agreed upon sequence in the *Backlog* queue. Now the *Backlog* queue represents the project portfolio of the company, in the best “negotiated” sequence, and presumably with the highest “business value” on top.

As before, the *Product Owner* of the first project will have the right to start; but from the second project onward, each *Product Owner* can decide when to start any one of their projects. Typically they will do so as soon as any of the front-line teams (A, B, and C) become available, because they are bonus driven.



Similarly, a *Product Owner* can “skip to the head of the line.” The front-line teams (A, B, C) respond on a “first come, first served” basis. Any individual *Product Owner* will obviously not change the sequence of his/her own projects, as they will be in some order of business value, which is reflected in their bonuses and tied to their best interest. (Remember, the bonuses are proportional to the monetary values of the projects too.)

So, it is clear: any *Product Owner* will be incentivized to skip the order on the initial table, as soon as a front line team becomes available to pick up work. A *Product Owner* can only give work to an idle team, because they have been instructed not to increase *Work in Process* unnecessarily.

However, to make the simulation even more realistic, this rule will not really be imposed; and *Product Owner* will front-load (because of their bonus drivers) the front-line teams. The front-line teams will end up having multiple projects in front of them (in their team *Backlogs*). The *Product Owners* might even “encourage” or “bribe” the teams to favor and start their own project before those of their colleagues! After all, this happens all the time in PEST environments, where internal power structures and politics always play a role.

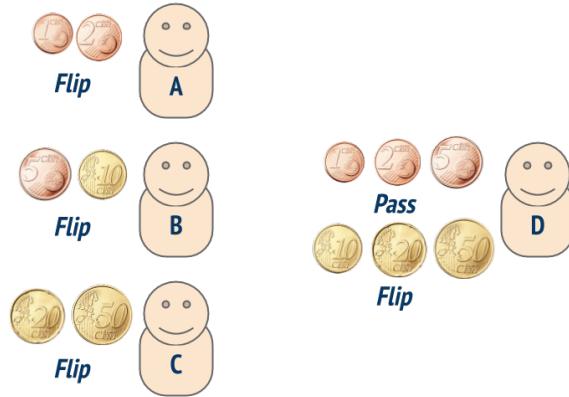
Before the project starts, we will ask the *Product Owner* of the *fifth* project to estimate its duration and due time and make them visible to all. Since we have not changed the rules, the *Product Owners* will continue to push work into the system.

All of this will create excessive *Work in Process*, and all project times will be longer - much longer! - than anticipated. What can be expected is that the fifth project will be delivered late with respect to the estimation provided, notwithstanding that the estimation was based on hard data and the actual *experience* of the earlier rounds. Sometimes empirical data is misleading, especially if the environmental variables that produced the data are not under known conditions or control.

Likewise, as in the previous runs, before the simulation starts, the participants are asked: “*Where do you think we might have a Constraint?*”

They will also be asked the same question several times during the simulation run, and at the end.

The Key Question is: Where is the Constraint?



Initially, the participants will convincingly reconfirm that think that the *Constraint* is Team D, due to the (apparent) greater *Work Load* it has.

As the simulation progresses, they will notice queues appearing in front of Team A. At the end of the simulation run, it should be obvious to all that the real *Constraint* is Team A.

They might not see or realize yet. The reason why will be explained in the upcoming chapter.

Takeaways

Simulations are used in teaching Agile and Lean. Such simulations are truly educational, and can provide many insights. Few of them however truly represent the chaotic situation that is the norm in PEST environments. Even less capture the dynamics of power structures, politics, and conflict of interests that typically happen in any reasonably sized company. These are all factors that perversely contribute to the degradation of operational performance and business results. The *TameFlow Simulation* tries to represent all these dimensions in a way that allows us to appreciate how they interact and - above all - give us insights about what to do about them.

Etienne Du Plooy says...

This is an excellent must read whether you know TOC, Kanban, Agile, CCPM, Six Sigma, and whether you've learnt the TameFlow approach or not. The book discusses Kanban in a Theory Of Constraints (TOC) context, and being a Throughput Accountant, what I really enjoyed was how Throughput Accounting (TA) measurements are used to keep business priorities in focus.

Many years ago, after learning TOC, I read David J Anderson's book, Agile Management, which opened and enlightened my thinking about Critical Chain Project Management. Now, with this "Tame your Work Flow" book, one gets additional clarity and learns what is required in multi project environments that have plenty of uncertainty to deal with. One needs to have simplified yet effective methods to achieve good systemic financial performance.

The TameFlow approach is focused and directly impacts where improvement needed. My biggest takeaway from this book is how to stay focused in PEST environments with TameFlow's Throughput Accounting measurements. A very good section discusses it and there are also good examples in the book.

Etienne Du Plooy, Theory of Constraints Jonah, Author of “*Throughput Accounting Techniques, Maximize the profit mix of your company.*”

11—Finding the Constraint in PEST Environments

In the simulation round described at the end of the previous chapter the *Constraint* in the *Work Flow* turned out to be Team A, rather than Team D.

How is that possible?

It contradicts the observations made during the first few rounds.

And it also defies logic: Team A will, on average, process one-third of all work, while Team D will, on average, will process one-half of all work (as the bronze coins do not require any effort by Team D).

So Team A should be doing *less* work than Team D (one third vs one half).

After the first couple of simulation rounds a queue formed in front of Team D, revealing that it was the *Constraint*. We were observing the effect of the *Constraint* in the *Work Process* of Team D.

However, when all ten projects were released into the system, a *different* major queue formed in front of Team A.

What was the cause?

In *Chapter 9 - Constraints in the Work Flow and in the Work Process* we learned that the location of a *Constraint* can move either because we elevated the *Capacity* of a *Constraint* (something that did not happen in the simulation, at least not so far); or because the random distribution of the *Work Load* (here encapsulated in the content of the bag of coins) changed - which is the case here.

In our simulation, while the participants were told that the coins were picked at random, in reality the set of coins was rigged. The facilitator of the simulation told a white lie, precisely so that the participants could go through the learning experience we are describing below.

Impact of the Shape and Form of Demand

The set of coins used to seed the ten projects actually had a predetermined distribution, as follows:

- Thirty **1c** coins
- Thirty **2c** coins
- Twenty **5c** coins
- Ten **10c** coins
- Five **20c** coins
- Five **50c** coins

While the participants might initially not appreciate having been led to believe that the set of coins was random, when in fact it contained a predetermined distribution, the situation is not unlike what you would experience in reality.

While we might not know in advance what kind of requests will come from our customers, we can be sure that the incoming *Work Load* will have a precise statistical distribution. That is what we mean by “*shape and form*” of the *Work Load*.

That shape and form (illustrated in the distribution table below) will be able to “flow” through the system in different ways, depending on how it hits the various *Work Processes* (i.e. the individual teams in the simulation).

At this point the participants are asked to reflect on how the knowledge about the coin distribution might change their understanding of what happened during the previous simulation runs. It is clear from the distribution that if there are thirty **1c** coins and only five **50c** coins, the team that is handling the **50c** coins will have much less work to perform than those handling the **1c** coins.

This predetermined coin distribution intends to simulate that the *Work Load* experienced by a multi-team system depends on how the *Work Load* will target the individual teams. While the *Work Process* of each individual team will not change, the shape and form of the incoming *Work Load* clearly affects how the work will flow through the system.

The *Work Load* is due to external causes. There is much more variability in external causes; most of it is in effect due to *Special Cause Variation*. This is where a *Kanban* mindset of focusing on queues becomes very important; though the signals that the *Work Load* is providing must not be masked artificially, for example by using *Column WIP Limits*. Otherwise we miss the information that is conveyed by any shape and form of the *Work Load* - and we further miss the opportunity to react and manage how the *Work Load* is released into the *Work Flow*.

A particular weakness of most *Kanban* implementations is that the queues in front of *Work Processes* (i.e. in front of the teams in the simulation) are not positioned in relation to one another. Without this fundamental step, the *Kanban Method* is unable to identify the *Constraint* in the *Work Flow*. (At most, the *Kanban Method* allows you to detect temporary bottlenecks *inside* a process that is modeled with *one Kanban Board*.) The inter-relatedness of the queues is lost - and actually distorted - by using *Column WIP Limits* within each *Kanban Board*!

Expose all (Virtual) Queues of Work Load without WIP Limit Distortions

Why is the inter-relatedness of queues in multi-team *Work Flows* so important?

To appreciate this question, we need to examine in more detail the “*shape and form*” of the incoming *Work Load* distribution as it hits the individual teams. We can see how this is the case in the table below. If the 100 coins from the “rigged” distribution are picked at random to seed 10 projects, then *one* possible *combination* (or specific instance of picked coins) of those ten projects *could* look like this:

One Possible Combination of the Coin Distribution per Projects

| Project | 1c | 2c | 5c | 10c | 20c | 50c |
|---------|----|----|----|-----|-----|-----|
| P01 | 2 | 1 | 2 | 3 | 1 | 1 |
| P02 | 0 | 5 | 2 | 1 | 0 | 2 |
| P03 | 5 | 1 | 2 | 2 | 0 | 0 |
| P04 | 4 | 3 | 2 | 1 | 0 | 0 |
| P05 | 5 | 2 | 3 | 0 | 0 | 0 |
| P06 | 3 | 3 | 1 | 1 | 1 | 1 |
| P07 | 4 | 3 | 1 | 0 | 1 | 1 |
| P08 | 3 | 4 | 2 | 1 | 0 | 0 |
| P09 | 2 | 4 | 3 | 0 | 1 | 0 |
| P10 | 2 | 4 | 2 | 1 | 1 | 0 |

Note well: we stress that the above is *one* possible combination of coins picked out of that very same predetermined distribution of 100 coins. Another combination might allocate a different tally of coins (in the various denominations) to the different projects and teams, but despite being a different combination, it comes from the same distribution (shape and form) of *Work Load*. It is the randomness of the distributions of the *Work Load* that will mostly determine where we will find and observe the *Constraint* in the *Work Flow*.

The viewpoint given by the above table is one that could be held by a top level manager or *Portfolio Manager*, outside of any single project or team within the projects. Their attention is on how much work goes *into the Work Flow*.

There is also another perspective that we must contend with at the same time: the perspective that one would have from within any particular project (like the perspective of a *Project Manager*) or from within any particular team working on the project (like the perspective of the *Scrum Master*).

In the *TameFlow Approach* - unlike any other - it is important to realize that when thinking about how to manage the whole system, we cannot selectively have either (the outside perspective or the inside perspective) viewpoint. We must necessarily have *both* in mind, and determine how *both* actually influence and shape the other.

Let us proceed to the next step and look at how the incoming distribution appears from inside the single projects or teams.

If we examine how the above *Work Load* translates to actual coin flips for the projects, then we will see this:

| Project | A | | B | | C | | D | | | Total |
|----------------|----|----|----|-----|-----|-----|-----|-----|-----|-------|
| | 1c | 2c | 5c | 10c | 20c | 50c | 10c | 20c | 50c | |
| P01 | 2 | 1 | 2 | 3 | 1 | 1 | 3 | 1 | 1 | 15 |
| P02 | 0 | 5 | 2 | 1 | 0 | 2 | 1 | 0 | 2 | 13 |
| P03 | 5 | 1 | 2 | 2 | 0 | 0 | 2 | 0 | 0 | 12 |
| P04 | 4 | 3 | 2 | 1 | 0 | 0 | 1 | 0 | 0 | 11 |
| P05 | 5 | 2 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 10 |
| P06 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 13 |
| P07 | 4 | 3 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 12 |
| P08 | 3 | 4 | 2 | 1 | 0 | 0 | 1 | 0 | 0 | 11 |
| P09 | 2 | 4 | 3 | 0 | 1 | 0 | 0 | 1 | 0 | 11 |
| P10 | 2 | 4 | 2 | 1 | 1 | 0 | 1 | 1 | 0 | 12 |
| Total per Team | 60 | | 30 | | 10 | | 20 | | | |

Team A is the constraint because it has the largest Work Load of coins to flip.

If we consider Project P01 on the first line of the distribution above, and the rules for flipping the coins, we see that the work that Project P01 has to undergo is as follows:

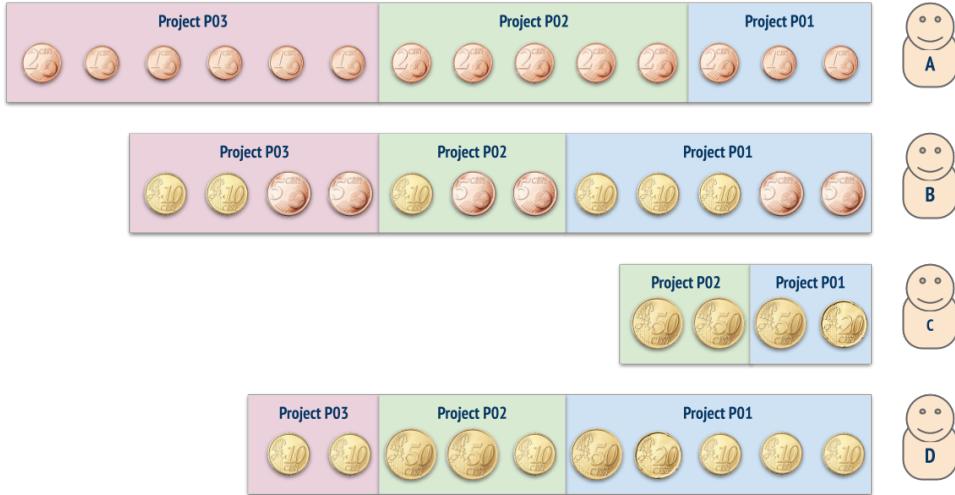
- Team A will flip two **1c** coins and one **2c** coins.
- Team B will flip two **5c** coins and three **10c** coins.
- Team C will flip one **20c** coin and one **50c** coin.
- Team D will flip the golden coins only, hence three **10c** coins, one **20c** coin and one **50c** coin.

In total, Project P01 will need fifteen coin flips, which represents the total effort to get Project P01 done.

In a similar way, we can count that Project P02 will need a total of thirteen coin flips; Project P03 will need twelve flips.

If we now reckon that the projects go through the process in order (because that was the business value rule we established), we can imagine that every team has in front of it a “virtual” queue of items in a *Waiting For* queue. The queue of each team will have its portion of work from each project - in order - joining the queue.

After starting just three projects, from the above allocation across the ten projects of the predetermined coin distribution, these “virtual” queues in front of the teams will look as follows:



Project 01 is the first one in the queue of each team; Project 02 is the second one; and so on.

After considering only three projects it is already evident that the queue in front of Team A is the longest.

After 10 projects, from the above *specific* distribution, the differences will be even greater. Here it is evident why the interrelatedness of *all* queues in the system becomes relevant to identify the *Constraint in the Work Flow*. It is clear from the above illustration, that the longest queue is the one in front of Team A, while in fact Team C is lightly loaded.

The *Constraint in the Work Flow* will show up in a *specific* location (i.e. in front of a specific team) depending on this *specific* coin distribution because the distribution determines the effort that the individual teams have to sustain. The team that is confronted with the largest effort is the *Constraint in the Work Flow*.

Note: Referring to the *Jeep, the Jungle and the Journey* metaphor you can imagine that the queues of coins are like a map of the *Jungle* while the four teams are our *Jeeps*. Once the teams start working and eat off the queues, that is our *Journey*. Naturally, there is also a parallel with the original *Story of Herbie*: the coins represent the number of steps remaining in front of each “scout” (the teams) for them to reach the base camp. Here “Jeep A” has the longest path to travel before reaching base camp!

Observe also how the 100 coins of this distribution are allocated across the ten projects will not change the location of the *Constraint in the Work Flow* (relative to the execution of all of these 10 projects). A specific coin allocation though might have an impact on the timing when the *Constraint* becomes relevant and to what extent it might be limiting.

This is particularly evident if you look at the table above, where in the last row we see the total *Work Load* per team. We see that this distribution will give the teams a *Work Load* as follows:

- Team A will have to flip 60 coins.
- Team B will have to flip 30 coins.
- Team C will have to flip 10 coins.
- Team D will have to flip 20 coins

Now it is evident that Team A is the busiest. Even if these 100 coins were allocated in a different combination to the 10 projects, the way the *Work Process* determines which teams do what kind of work will set up *Team A* as the *Constraint* in the *Work Flow*, relative to this *specific* coin distribution.

As the shape and form of the incoming work changes, then the location of the *Constraint* in the *Work Flow* can change; and we need to be quick to update our *Work Release* policies. We can do that by learning more about *Drum-Buffer-Rope* scheduling and how to manage the *Constraint* in the *Work Process* of the particular team that happens to be the *Constraint* in the *Work Flow* at the moment. More about these two topics in the next two chapters.

Note: For the sake of clarity, the above illustration does not mean we are trying to make all work processes end at the same time. The alignment of the queues along the right side of the picture is simply a visual trick used to show how much work is *in front* of each team *relative* to the work that is in front of the other teams. Moreover teams A, B and C still need to process their work before Team D enters into action; but in order to identify the *Constraint* in the *Work Flow* we are not concerned with timing, sequencing, prioritization or parallelization of work streams. We only need to determine *how much* work is going to hit every team, and focus in real time on the one that will get the highest *Work Load*. With this visualization of the “virtual” queues of *Work Load* hitting every team, we can effectively reason in terms of *Constraints Management*, and perform Step 1 of the 5FS: *Identify the Constraint!*

With this new insight - that Team A is the *Constraint in the Work Flow* - the strategy becomes to find the *Constraint* in the *Work Process* of Team A.

We will be concerned about the *Constraint in the Work Process* of Team A, and only of Team A, because it is clearly the *Constraint* in the overall *Work Flow*. If, at a subsequent phase, the *Constraint* moves to a different team, then we would focus on the *Constraint in the Work Process* of that team, and no longer be concerned about the *Constraint in the Work Process* of Team A.

Lesson Learned: Use the *Constraint in the Work Flow* to identify the team where we need to focus on the *Constraint in the Work Process*.

Once we know where the *Constraint in the Work Flow* is located, we can then manage the *Work Release* into the system according to the *Drum-Buffer-Rope* practices as described in the next chapter.

Moreover, from the above illustration, it is also evident that Team C is the least loaded. Therefore (if skills and capacities allow for it) members of Team C should try to help out Team A which is the most loaded.

Lesson Learned: In order to identify the *Constraint* in the *Work Flow*, it is necessary to observe mathematically the real workflow as it happens.

This is the motivation for having a *Waiting for* queue in front of each and every team. Queue size is a quantitative metric and a leading indicator of performance decay. This queue will render visible the *Work Load* hitting every single team.

You might consider these *Waiting for* queues similar to the team *Backlogs* used in Scrum.

It is also the justification why we should avoid *Column WIP Limits*, because such limits will distort the queues and remove the *Work Execution Signals* that we can read by observing the queues.

Note: While, in the described scenario, Team A is the *Constraint* in the *Work Flow* and we decided to focus on its *Constraint* in the *Work Process*, it is important to be aware that the *Constraint* can move. We know this already. It can move because we might elevate the *Capacity* of the *Constraint*. We also just learned that it may move because the incoming “shape and form” of the *Work Load* changes, and moves the *Constraint* in the *Work Flow* to some other place. What we don’t know yet, and we will more fully discover in Chapter 19 - *Execution Management in PEST Environments* and in Chapter 20 - *Operational Governance in PEST Environments*, is that the *Constraint* in the *Work Flow* can also move because of events that happen during the *Work Execution*.

In order to perform step one of the *Five Focusing Steps* - i.e. Identify the *Constraint* - we must be able to see all work in front of every team. Making the *Work Load* visible and/or measurable is essential. Unless the actual *Work Load* of each team is made visible, we will be unable to apply the principles of *Constraints Management*.

Lesson Learned: A *Waiting for* queue must necessarily list all work that the teams have to perform; even work that ordinarily might not be captured in a product backlog.

This means that even “trivial” or “obvious” tasks and activities need to find their place in these queues. We never want *any* work to be *invisible*. This is for the technical aspect of being able to properly identify the *Constraint* - but it also has the invaluable side effect of making things very transparent. All teams should acquire the habit of always making their part of work visible.

Focus on the *Constraint* in the Work Flow First, then on the *Constraint* in the Work Process

The distribution of the coins determines where the *Constraint* in the *Work Flow* will appear; the relative *Work Loads* that hit the various teams have a huge impact. Once a team is identified as the *Constraint in the Work Flow*, we will look into that team's process and think in terms of the *Constraint* in the *Work Process* of that specific team.

All other teams will also have *their Constraint in their Work Process*, but it is irrelevant to be concerned about them, because the corresponding teams are not the *Constraint* of the overall *Work Flow*. Any improvements on those teams is moot.

This is also why every team should be equipped with *Drum-Buffer-Rope (DBR) Kanban Boards*. Only the team that is the *Constraint in the Work Flow at the moment* will actually activate the DBR features of its *Kanban Board*. All other teams, while being instrumented and collecting metrics to work with a *TameFlow DBR Board* or a *Throughput Management Board*, will not employ it. They will just use the *Flow Efficiency* boards described earlier and keep the DBR features on standby, ready to be mobilized *ad hoc*, as soon as the queue signals in the *Work Flow* indicate that their specific team is the *Constraint in the Work Flow*.

Lesson Learned: Only the team that is the *Constraint in the Work Flow* should be concerned about managing its own *Constraint in the Work Process*. It is the only team that must use the *DBR* or the *Throughput Management Board* (while the other teams can use a *Flow Efficiency* board, and keep ready to activate the DBR feature should they become the new *Constraint in the Work Flow*).

With this instrumentation, we can use the ideas of *Constraints Management* even in the case of extreme VUCA scenarios, because the system will be very reactive, and we can detect where the *Constraint* lies in the overall *Work Flow* - without the distortion and interference induced by *Column WIP Limits*.

Demand varies and fluctuates in the real world. Therefore it is essential to have a highly reactive system that we can easily be shaped with the proper instrumentation and managed with a shared understanding across the entire organization.

Note: In *Chapter 19 - Execution Management in PEST Environments* and in *Chapter 20 - Operational Governance in PEST Environments* we will also learn how to make this system even more reactive by being able to detect the ephemeral *Constraint in the Work Execution* - which, when detected, would take precedence over the *Constraint in the Work Flow*.

Likewise, it becomes clear why any kind of *Demand Shaping* (i.e. the deliberate determination of the *Work Load* distribution) is so important because the more stable we can make the incoming *Work Load* distribution, the easier it is to keep the *Constraint in the Work Flow* in the same place. With stability comes predictability and performance.

Lesson Learned: The *Constraint* in the *Work Flow* is determined (primarily) by the “shape and form” of the incoming *Work Load*. The *Constraint* in the *Work Flow* is detected through *observation* of incoming *Work Load* queues (and also through *analysis* of the recent past performance of every team, one with respect to the other). The *Constraint* in the *Work Process* is determined (primarily) by *Common Cause Variation* and is identified through *Flow Metrics* at the process level. The *Work Flow* is the *Jungle*. The *Work Process* is the *Jeep*.

Note: Observation of incoming *Work Load* queues requires estimation and/or forecasting. In *Chapter 13 - Portfolio Prioritization and Selection in PEST Environments* and in *Chapter 18 - Full-Kitting as Ongoing Executive Activity*, we will learn how this can be done without requiring too much “up-front” work that would be considered contrary to Agile Principles.

Takeaways

Finding the *Constraint* in PEST environments is not straightforward. When multiple teams are part of a value delivery network, through which all work flows, the *Constraint* in the *Work Flow* is the team that is facing the longest queue of *Work Load*.

However, since that team can be deep inside the value network, it is not always evident that it will be the *Constraint* just by looking at the *Work Load* that the team is facing at the moment. Instead we have to look at the *virtual queues* that will be determined by the shape and form - the statistical distribution - of all known incoming work.

Once we know which team is the *Constraint* in the *Work Flow* then we can focus on the *Constraint* in its *Work Process*.

Hermann Hyytiälä says...

I think this book is what we really need after Agile, TOC and Kanban to improve the effectiveness of knowledge-work in our organisations.

Hermann Hyytiälä, Keynote speaker, writer, thinker, Principal Consultant at Gofore.

Sanjeev Gupta says...

This is a serious book, for organizations and managers serious about improving their software delivery or DevOps processes. Not an easy read, but so what!

Sanjeev Gupta, TOCICO Lifetime Achievement Award winner, Founder and CEO of Realization Technologies.

12—Drum-Buffer-Rope Scheduling

At the end of the previous chapter we gained the insight that in order to detect the *Constraint* in the *Work Flow*, the work needs to flow *unobstructed* through the system, as we are moving from pull to flow. It is only by *observing* the queues forming in real time (or even better, being able to forecast where they will form - although that is another topic) that we will be able to apply the ideas of *Constraints Management*. Quantitatively measuring and visually observing the size of queues is a valid scientific approach.

In this chapter we will see how to deal with *Portfolio Management* of multiple projects being routed to multiple teams in a VUCA environment. It is all about developing sharp reflexes and be responsive when features, epics, projects and *Work Items* run through our value stream or value network. To be able to represent and visualize this multitude of moving parts over a number of coordinated boards is a key skill one must possess.

Reflections on *Column WIP Limits*

If we need the *Flow* to be unobstructed it does not make any sense to introduce work state, i.e. “*Column WIP limits*” as it is strongly suggested by the *Kanban Method*. Such *Column WIP Limits* interfere with the *Flow* in an artificial way, and they will actually prevent us from detecting the critical leading signals we need.

Mind you, the *Column WIP Limits* proposed by the *Kanban Method* have a rationale: they serve the purpose of *limiting* the *Work In Process* as well as providing the possibility to react to *Special Cause Variation* - which *may* affect the *Work Processes* visualized by the *Kanban Boards*.

However, *Column WIP Limits* have many negative effects that impact both the individual *Kanban Boards* as well as the more convoluted *PEST* scenarios with several concurrent streams of work which are being examined here.

In short, *Column WIP Limits*:

- Hide the real *Constraint*.
- Give too many false positives of problems that are not really problems.
- Promote excessive trial and error leading to system tampering.
- Changing *Column WIP Limits* is a form of tampering.
- Loss of focus (attention is deviated away from the *Constraint*).
- Promote local optimizations.
- Waste “improvement” efforts when they target the *non-Constraint* columns .
- Introduce artificial bottlenecks.
- Hinder and disrupt true *Flow*.
- Promote directionless evolutionary change.
- Cause the “moving *Constraint* syndrome.”
- Lead to improvements by accident and happenstance, that happen to involve the *Constraint*; and not by deliberate and targeted action on the *Constraint*.
- Induce system instability, making it less predictable and reliable.

An important function of *Column WIP Limits* is to facilitate the *pulling* of work. The conventional rule is that if any column contains less *Work Items* than its *Column WIP Limit*, then the worker(s) of that process step can *pull* work from the previous state. This *Pull Signal* is propagated “backward” until the beginning, to the left most column of the *Kanban Board*, where it becomes a *Replenishment Signal*, which signifies that more work can be released into the system.

Notice though, that in the simple scenario of the simulation, there is no one *single* “leftmost column” - because we have three teams that are all located at the beginning of the *Work Flow*. So this simplistic pull replenishment policy cannot work.

What would happen if Team C (which we know from the analysis we did in the previous chapter) has spare capacity and wants to pull work into the *Work Flow*? A new project would start, and - yes! - Team C could take it on; but the same project would be stuck in the queue which is naturally building up in front of Team A (because as we know, Team A has the highest *Work Load*).

Even worse, when Team C will have finished processing that new project, it would be passed onto the next team in the *Work Flow*, Team D; but Team D wouldn't be able to handle that work until it receives the corresponding deliverables (coins) from both Team A and Team B. So that project brilliantly started by Team C, would be at a stand still in front of Team D until the corresponding work is delivered much later by Team A and Team B.

The virtual queue in front of Team D would be growing.

Note: When using conventional *Kanban Boards*, the situation becomes absurdly bizarre. Team D wouldn't even notice that there is a growing queue in front of it, because that queue would materialize in the *Done* hand-over column at the end of Team C's board, and not in a *Waiting For* column at the beginning of Team D's board. Team C would happily pat themselves on the back believing they are very productive, maybe complaining about Team D's laggards; all the while Team A would continue with no notion that they are the source of the slowdown. Furthermore, if awareness is gained, then the entire system would go into a frenzy trying to understand how to manage the “dependencies” between the three teams. In reality there are no dependencies to manage; one only needs to know where the *Constraint* is and apply the 5FS.

In short, the replenishment signals that are typical of the *Kanban Method's Column WIP Limits*, would have the atrocious effect of increasing the amount of *Work in Process* as both queues, in front of Team A and D, would grow, increasing *WIP* with all the known dire consequences this has in a system.

Note: Conventional *Kanban Boards* would not even recognize this as *WIP*, because it would be classified as work that is in the teams' *Backlogs*, and thus not even started. But we can see now that it is truly *WIP* when we raise from the team perspective and look at the portfolio of projects going through the *Work Flow*. Paradoxically, in a PEST scenario, the *Kanban Method's Column WIP Limits* defeat their very purpose: that of limiting *Work In Process*!

Note that in the team configuration of the example, if there were another team downstream of team D, it would be starving as it would not receive any work from Team D (though that is not necessarily bad,

since it would not be a constraint).

Actually, *Team D* would be the first starving team despite the fact that it would have a full load of work in front of it!

This observation exposes another paradox of *Kanban Boards*: teams can be starving even with queues in front of them!

The way to prevent this from happening is by limiting *Work in Process* as Taiichi Ohno first taught: i.e. that by continually lowering the amount of work in the system, so that at the end the only resource that would be working while all others would be idle would be the *Constraint* - and it would be working all the time.

Practitioners of the *Kanban Method* try to apply the spirit of this precept, but fail in practice by resorting to *Column WIP Limits*. It works, as long as the context is simple and linear but fails in presence of *PEST*.

The best way to limit *WIP* in the presence of *PEST* is not via *Column WIP Limits* but simply by letting the *Constraint* determine how much work can actually flow through the system in order to regulate and minimize the amount of *Work In Process* according to the *Capacity* of the *Constraint* itself.

The problems can now be stated:

1. How can we *limit Work in Process* without using *Column WIP Limits*?
2. How can we *Postpone Commitment* to the very last responsible moment without disrupting delivery all the while keeping the amount of *Work in Process* minimal?
3. What *Replenishment* policy should we use, and what *Replenishment Signal* should trigger it?

The answers to follow!

Introducing Drum-Buffer-Rope in the Work Flow

These problems have been addressed squarely by both *Critical Chain Project Management* and by the *Drum-Buffer-Rope* (DBR) scheduling rules of the *Theory of Constraints*.

Note: While we introduce the concept of *DBR* here, it will be explored in more detail later. So you might want to revisit this chapter again after reading the next few chapters. This reversal of logic is unavoidable because in our exposition we are looking at the big picture (the *Work Flow*) before we delve into the small picture (the *Work Process* of any individual team); and *DBR* scheduling will be explained more effectively with the latter, in *Chapter 14 - Flow Efficiency, DBR and TameFlow Kanban Boards* and then considered again for the *Work Flow* in *Chapter 18 - Full-Kitting as Ongoing Executive Activity*.

Some aficionados of the *Kanban Method* like to point out that the story with Herbie is in a linear paradigm and in no way survives queueing theory where you just need to add more queues to resolve any issue. Dr. Goldratt had a PhD in physics and surely knew of queueing theory - theory dating from as far back as the early 1900's. But be that as it may, and put as many queues as you wish, Herbie shall remain the *Constraint*. Until you address and manage the *Constraint - Kanban Method* or not - the *Constraint* will affect you.

In the simulation, participants are introduced to the concepts of *Drum-Buffer-Rope* scheduling and then invited to put it into practice. An important policy is also instituted: the *Product Owners* are not allowed to skip to the head of the line. Instead, the projects must be released into the work flow in the

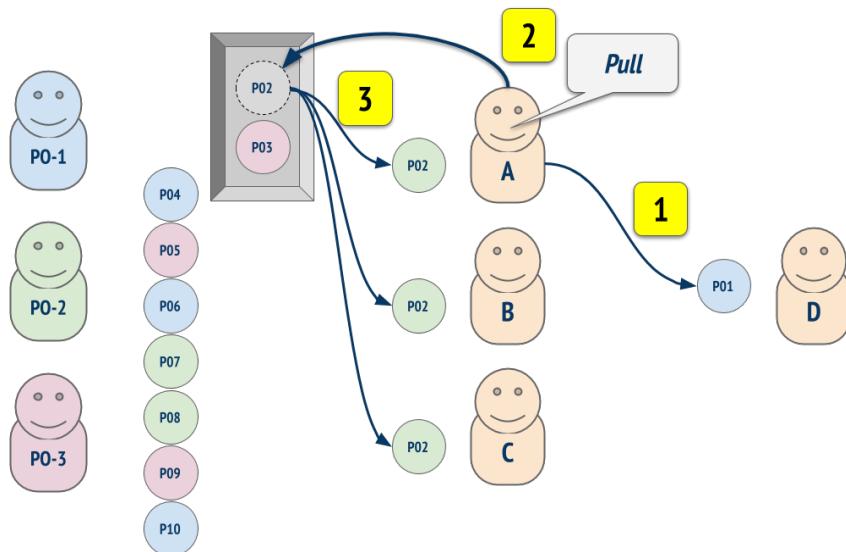
order that was established from the outset (with the negotiation or competition between the *Product Owners*.)

Note: It takes a deliberate top management decision to adopt *DBR Scheduling*. The *Mental Models* presented in the first few chapters need to be well understood. In most companies there is simply not sufficient *stability* to manage queues in this fashion. Typically they chase bottlenecks and address it directly; but we know better now that this is not the smartest way to proceed. Likewise many Agile practices might cancel out important differences (i.e by “resetting” at the end of each iteration), effectively making it impossible to detect those differences in capacity that allow us to identify the *Constraint* and then move over to *DBR Scheduling*.

Of course, the above mandates that the *Product Owners* really play the game according to the rules. Yet they have to contend with their self-interested drive to skip to the head of the line and increase their odds of being rewarded with their bonuses. We can expect them to be extremely unhappy about this rule; but in the next chapter we will see how to resolve their conflict of interests in a way that is beneficial for the whole system.

Drum-Buffer-Rope in Practice

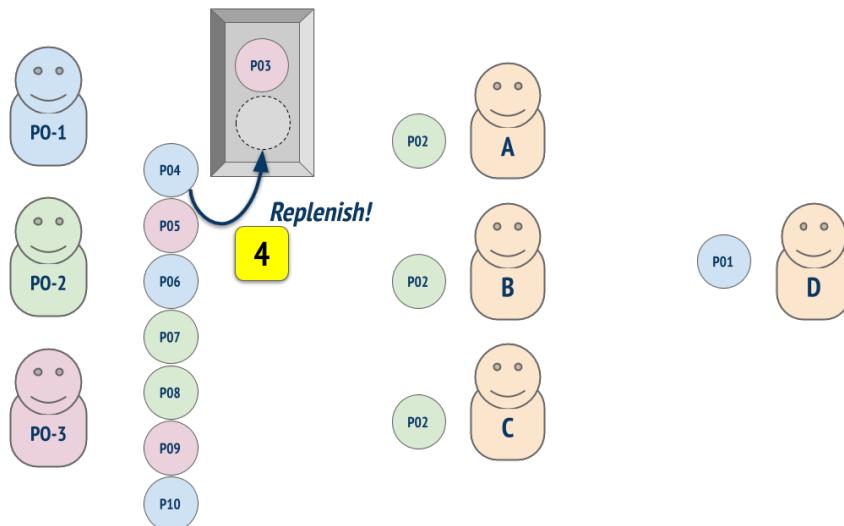
A *Buffer* is placed in front of the *Constraint* in the *Work Flow* (Team A), with provision, say, for two projects.



Here is how this works:

1. Suppose Team A has just finished working on project P01, and has handed its coins over to Team D, and hence has the capacity to take on the next project.
2. Team A then will raise the *Replenishment Signal*.
3. Team A will actually pull its part of the next project, project P02, from the *Buffer* into its part of the *Work Flow*. At the same time Teams B and C will react on the pull signal and pull their part of project P02 into their part of the *Work Flow*.
4. As soon as P02 leaves and frees an empty spot in the *Buffer*, the *Buffer* will be replenished with the top most project from the *Backlog*.

The evocation one can have in mind here is that Team A sets the pace and gives the *Drum* beat by raising the *Replenishment Signal*. When the signal reaches the *Buffer*, it acts as the *Rope* which pulls in work into the *Work Flow*. In practice, those teams that should pull work will do so when they hear the drum beat. The drum beat is like the cry of a baby that signals it needs to be fed!



The idea is to always have a minimum amount of projects in the *Buffer* which are ready to be pulled into the *Work Flow* by the *Constraint* (Team A) at any time. This way we ensure that the *Constraint* never runs out of work, and executes at maximum capacity.

Note: Lost Capacity on the *Constraint* (Team A) is money foregone forever.

This will mean that all the other teams, which all have *more Capacity* (or *less Work Load*) than Team A, will at times stand idle waiting for work.

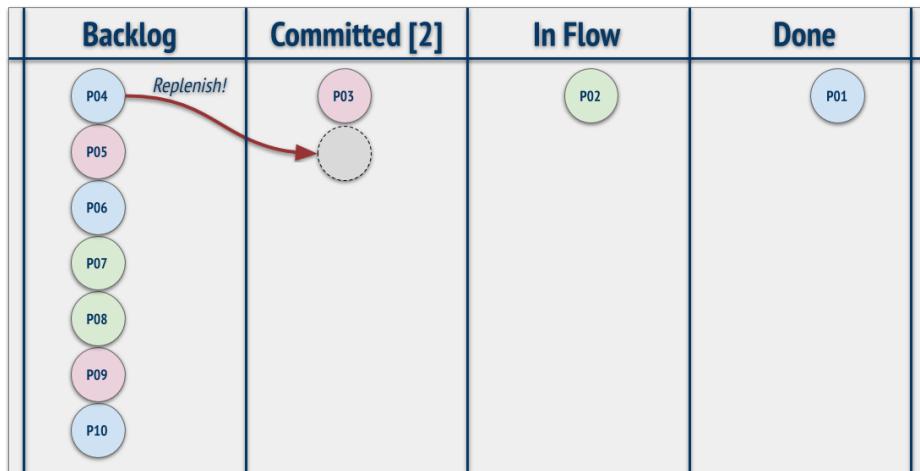
Note: This is a key element of high performing organizations: workers will be waiting for work, rather than work waiting for workers. Focus on emptying the queues, not on keeping the (*Non-Constraint*) workers busy!

When the simulation is run with the new setup, the participants are asked to estimate the duration and the forecast completion time of the fifth project. It is quite likely that this run will be the best delivery of that fifth project, in terms of duration, timely delivery and possibly earlier delivery compared to the previous runs because of the queuing discipline imposed by the *DBR Scheduling*.

The integration chaos previously witnessed on Team D; the clocking of work; the assembly of coins to the proper project; will also feel the positive impact of the new system.

Drum-Buffer-Rope Portfolio Kanban Boards

If we take the second illustration above, and try to visualize the situation through the *Kanban Board* below we start to get an embryonic *Portfolio Board* - a board where the *Work Items* represents projects or initiatives in a *Portfolio* - that looks as follows:



Notice that the *Buffer* is represented with the *Committed* column in front of the actual work column. The “work in progress” column itself is called the *In Flow* column, to emphasize that it is where the “flowing” of the work happens, stressing the idea that the work has to “flow” rather than being “in progress.”

The *Buffer* column has been given the name of *Committed* (rather than “*Buffer*”). The intent is to show that the decision to execute the projects that are placed in the *Buffer* is irrevocable.

Note: As we will observe in a moment, the *Committed* column on the *Portfolio Board* is a different concept than the *Buffer* of the *DBR Scheduling*. It just so happens that for the moment it is convenient to consider them as the same, for ease of explanation - but do keep in mind that they are two distinct elements.

This, in turn, means that the projects in the *Backlog* are to be considered as *options*.

So we start to form the idea that the projects in that *Buffer* are committed to, while the other projects in the *Backlog* are options - and this in turn gives us the flexibility to change the projects in the *Backlog*, reorder them, or simply take them out. As we will see, this is a very important characteristic of how we approach *PEST* control.

Location vs. Visualization of the *Constraint* in the *Work Flow*

The mapping between the illustration of the *Work Flow* and the *Kanban Board* is not perfect, in the sense that we illustrate a situation where the *Buffer* placed *in front* of the *Work Flow* is to ensure that the *Constraint* process is never starved for work and is not merely a visual artifact.

Of course, the *Constraint* could be further downstream in the *Work Flow* than the frontline Team A. For example, at the beginning of the simulation it appeared that the *Constraint* was Team D. In fact, for a different distribution of coins, Team D could very well be the *Constraint* in the *Work Flow*. In such instances, the *Buffer* column should be placed *inside* the *Work Flow* process, just in front of Team D, the *Constraint* process.

However, for the sake of illustration and to make it clearer, we place this column just before the *Work Flow*. Be aware that it could be anywhere inside the *Work Flow*.

That way the *Replenishment Signal* is clearer and more visible on the *Portfolio Kanban Board*.

Even if the *Buffer* were placed further downstream to protect the *Constraint* at the position where it actually appears in the *Work Flow*, from a conceptual level, the *Replenishment Signal* comes from the system and is directed to the *Backlog*. Therefore, it is convenient to imagine the *Buffer* as a “box of commitments” *just ahead* of the *Work Flow* and just after the *Backlog*.

Note: The act of “committing” to a piece of work is different than the act of putting that same piece of work in a *Buffer* that is protecting the constraint. It must be clear that in this particular board design, the *Committed* column also acts as (a proxy for) the *Buffer*. Keep in mind that if the *Constraint* team were not a frontline team, then we would more realistically design a board splitting the *Committed* column in two: the *Committed* column proper, that remains where we have seen here, and a *Buffer* (*Waiting for...*) column in *TameFlow Board* of the specific team that is the *Constraint* in the *Work Flow*. More about this

in Chapter 14 - Flow Efficiency, DBR and TameFlow Kanban Boards.

The Minimum Number of Work Items in the Buffer

Observe that the *Committed* column header also contains a number within square brackets. While such a number might remind us of a *Column WIP Limit*, this number is not a *Column WIP Limit*. It does not represent the *maximum* number of items that can be present in the column. Instead, it is there to remind us of the *minimum* number of items that should be present in the *Buffer* at all times to keep the *Constraint* busy.

Note: It is worth repeating: the logic is the *opposite* of *Column WIP Limits*. *Column WIP Limits* act as space buffers in a “*do not push*” construct in the sense that as long as there is no “space,” you are not allowed to let new work enter. With DBR scheduling, a Kanban system becomes a true “pull system.” When the *Constraint* finishes a task, a new one is allowed to enter the system. The logic of such a *Buffer* is in fact based on timing, not on space. While the *Buffer* is visually represented with space on the board, it is really a *Time Buffer*. Ideally it is sized so that replenishment from the start of the *Work Flow* to the *Constraint* happens just-in-time to guarantee that the *Constraint* is never starved of work.

Any lower number of items should be a matter of concern or even alarm. When there are fewer items, they need to be replenished. (As we will see in the next chapters, there are further elements of control that will be derived from the number of items present in the *Buffer*.)

The actual size of the *Buffer* should always reliably ensure that the *Constraint* process will never run out of work.

If the *Constraint* is way downstream in a long *Work Flow*, it will require a larger *Buffer* because it must take into account that the work has to traverse the long upstream *Work Flow*, which takes time.

Conversely, if the *Constraint* is at the beginning of a *Work Flow*, the *Buffer* can be smaller - and for the same reason: the time it will take for a *Work Item* to reach the *Constraint* process will be shorter, because there is less space to cover. In other words, the length of the rope and the size of the *Buffer* are related.

Lesson Learned: DBR Scheduling will limit the *Work in Process* without using *Column WIP Limits*, and it will do so by reducing the actual *Work In process* to the absolute minimum necessary to keep the *Constraint* busy at all times while the *Work Flow* progresses smoothly.

Lesson Learned: The *Backlog* is considered as a prioritized list of *options*, while the *Constraint*'s *Buffer* contains the minimum amount of *Work Items* that has been committed to.

Lesson Learned: The pulling of *Work Items* from the *Buffer* will be interpreted as a *Replenishment Signal* which must trigger releasing into the *Work Flow* the next topmost *Work Item* of the *Backlog*.

The **Constraint** in the Work Process of the Constraint in the Work Flow

With *DBR Scheduling* limiting and governing the overall *Work Load* that gets released into the *Work Flow* of a complex *PEST* environment, we will achieve numerous operational benefits.

Yet we can do even better! Now that we know how to find the *Constraint* in the *Work Flow* (Team A in our example), we can focus on the *Work Process* employed specifically by that team, and manage it in the best way possible. However, in order to do that, we need to learn about the *TameFlow Kanban Boards*, which is the topic of *Chapter 14 - Flow Efficiency, DBR and TameFlow Kanban Boards*.

Takeaways

We started this chapter reflecting on the limitations (no pun intended) of *Column WIP Limits*. Their intent is both to detect “bottlenecks” and to *limit WIP*.

We discovered that limiting *WIP* can effectively - and alternatively - be realized through *Drum-Buffer-Rope Scheduling*, and that we can achieve that in practice by using an appropriately placed *Buffer* column.

The *Buffer* column is conceptually set in front of the team that is the *Constraint* in the *Work Flow*. But when we are looking at the *Portfolio* board, in order to keep the view as linear and as simple as possible, we make the *Committed* column act as a *Buffer* for the entire workflow too. (This is a simplification that helps us run the *TameFlow* simulation more easily too.)

At a more detailed level, the *Buffer* column should really be the column within the *TameFlow Board* of that team; specifically the column that corresponds to the ‘Waiting for...’ semi-column of the step that is the *Constraint* in the *Work Process* of that team. Operationally that is the column that should trigger the actual *Replenishment Signal* to pull in work.

When the signal reaches that team’s *Backlog* column and pulls in the last item of a project, the *Replenishment Signal* is propagated further to the *Portfolio* board, where the next *Project* is pulled from the *Committed* column and released into the *Work Flow*, and distributed to all front line teams.

The details of how the *Portfolio* board interoperates with the team boards will be seen in more detail in *Chapter 14 - Flow Efficiency, DBR and TameFlow Kanban Boards*.)

Matthew Croker says...

With "Tame your Work Flow" the authors have created a spicy yet refreshing combination of Goldratt's Theory of Constraints and Toyota's Kanban.

Through this book, one can quickly appreciate the effectiveness of Kanban in its simplicity and the power it gains when paired with the Theory of Constraints, not only with a beauty of theoretical and philosophical harmony but most especially with the practical implications of the tools presented.

Knowing that Kanban is an approach that can be used for practically every existing domain requiring some degree of organization (from car manufacturing to organizing your next vacation), the book has thrown open the doors of my imagination on the possible applications of the Theory of Constraints through the TameFlow Approach.

Matthew Croker, Agile Coach, Co-creator of the Decision Espresso method, founder of faceofscrum.org

13—Portfolio Prioritization and Selection in PEST Environments

In the previous chapter we learned that *DBR Scheduling* is the best way to regulate and minimize *Work in Process*. Work should be released into the system at the same rate as the *Constraint* is able to absorb it. Release anything more and the *Constraint* will be overburdened (*Work in Process* will increase, *Flow Efficiency* will deteriorate, queues will appear in several places in the *Work Flow*, and average *Flow Time* will rise). Release anything less, and the *Constraint* will be waiting (or starving); and any time lost on the *Constraint* is a permanent loss for the system's *Throughput* - both operational and financial.

If *Work Items* have to be released exclusively when a *Replenishment Signal* is received from the *Constraint*, it is obvious that we need to handle one of the most subtly disruptive forces that is at work here - the conflicting interests of the multiple stakeholders who are represented in the simulation by the different *Product Owners*.

Every stakeholder will want their projects to be finished as soon as possible; and if that hurts another project, they could not care less. Both the stakeholders (i.e. customers) as well as the *Product Owners* would like to take advantage as much as possible of the resources in the system. Customers want to get their deliveries sooner rather than later. *Product Owners* have bonuses and budget allocations tied to their due date performances.

The problem that must be addressed is how to avoid having *Product Owners* skip to the head of the line, and try to feed their projects into the system as soon as possible, in the hope to get them back out of the system faster. We know that this behaviour increases *Work in Process* and at the end will make *all* stakeholders lose even more than what they hoped to gain.

If the *Replenishment Signal* indicates that *one more Work Item* can be released into the *Work Flow*, the immediate next question is: *Which Work Item should we pick?*

From a business perspective, it should be the item that provides the highest business value. The consequence is that the incoming stream of requests needs necessarily to be prioritized in some sequence of decreasing business value.

The *Product Owners* of course need to be informed that the *Replenishment Policy* has changed accordingly; but this also needs to be followed up with a change in how bonuses are recognized and how budgets are allocated. Instead of being measured on how soon they *individually* deliver their projects, the *Product Owners* will be rewarded based on how they *collectively* maximize the business value produced for the company. They need to be put in a situation where they all share the same destiny, and develop a sense of *Unity of Purpose*.

In practice, there needs to be a way for them to *collectively* produce an *agreed upon prioritization* of all incoming work; to *pre-empt interruptions* and skipping of lines; to execute *cooperative decision-making*. And if this does not work, there must be shared policies by which any unresolvable conflict is quickly escalated to upper management for immediate and indisputable resolution; which in turn implies working with how *information flows* within the organization for the best achievement of the company's *Goal*.

Once the work is released into the *Work Flow*, it must be understood that *Multitasking* and parallelism is not admissible; it must also be appreciated that the *Work Load* must be sustainable for the *Constraint* (and consequently any other team as well) in order to ensure an optimal, uninterrupted *Operational Flow*.

If there are problems arising during prioritization or later when work is in process inside the *Work Flow*, there must be explicit and rapid escalation policies with guaranteed response from higher management levels. The information that reaches the higher levels of management must arrive just in time and be the

minimal required to make informed executive decisions. The design of the *Work Flow* must be completed with a design of *Information Flow*.

What makes the situation even more challenging are multiple *Events* - typically deadlines. There is only so much work that can be done within a given time frame. Therefore, it becomes critical to decide which work we should *not* do; and to be very selective of what we decide to do instead. *Portfolio Management* is all about what you decide *not* to do.

Let us see how we can learn how to do this by going back to the Tameflow simulation.

WARNING! We must emphasize that the situation described hereafter comes from the *TameFlow Simulation*. There are many simplifying assumptions for the purpose of allowing the participants (and readers here) to easily build a *Mental Model* in order to understand a fundamental concept - specifically that of *Throughput Rate* - and how it makes sense to use it for prioritizing incoming work. When doing this in the real world, the equations of *Throughput Accounting* would be used, rather than tally up coin flips. So if you already happen to be familiar with the conventional way of doing this in *Throughput Accounting*, bear in mind that the purpose of this chapter is propaedeutic to *discover* a concept; and not a cookbook on how to use the concept in practice.

Prioritizing for Business Value

In a promotional drive, the marketing department has decided that all projects will be sold for the same price. Now, they are each worth 100 million Euro.

OK - this is a trick!

It serves to “normalize” all projects and make us reflect on how to *create value* by deciding how to *synchronize, coordinate, prioritize* and *select* the most valuable projects. If all the projects are worth exactly the same, then in theory any sequence will do.

Or would it?

Actually a huge difference can be made by how well the organization executes the individual projects. And because they will engage the system differently, and in particular they will engage the *Constraint* in the *Work Flow* differently, we have a way to leverage all decision-making around how efficiently the *Constraint* in the *Work Flow* can produce value for the business.

Note: We must stress that all projects are rigged at 100 million Euro business benefit precisely to make us appreciate how we can create more value by making better rational and profitable operating decisions. If the projects had different price tags, it would be too easy to make a biased and most likely wrong decision, just as it happens all the time in the real world. By artificially setting the value at 100 million Euro we are forced to *think!* That is what *Mental Models* are all about.

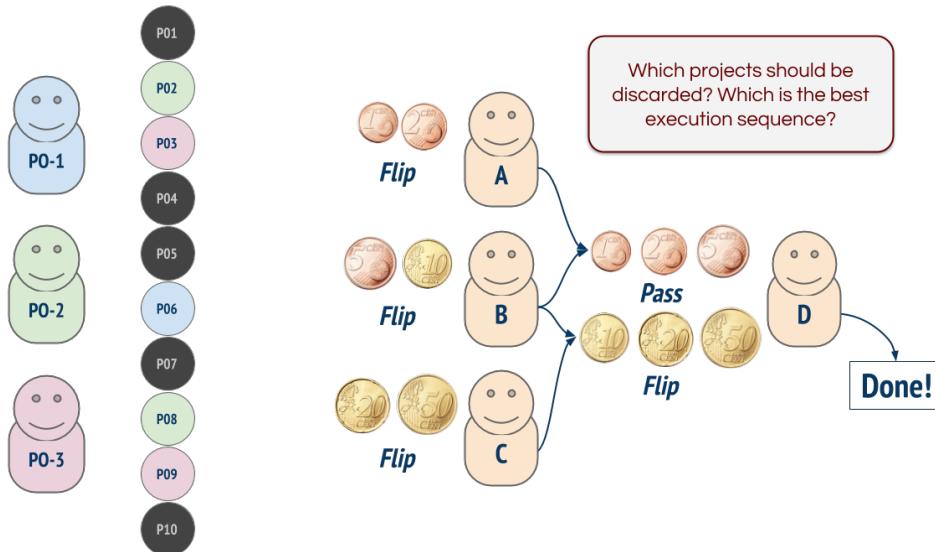
Furthermore, we have a deadline!

The accounting department needs to close the books before the end of the year. Given the timeline, the organization will be able to deliver only five out of the ten projects.

The question then becomes: which are the top five projects that will deliver the most bang for the buck? Five out of ten projects must be discarded. Which ones? And in what particular order?

Remember, nominally all projects are worth the same: 100 million Euro. So can we make a difference just by deciding which projects should be undertaken, and even more by executing them in a certain priority rather than another? If one project turns out to be more valuable than another, we want to get it delivered faster because if we did it differently - and something went catastrophically wrong - we might miss out on delivering that valuable project since we were focusing on something less valuable.

Our overall scenario unfolds as illustrated in the following picture. Five out of ten projects (in black) must be excluded, and the remaining five need to be put in a priority order that maximizes the value for the business.



While there are many ways project teams decide on prioritization order, often it ends up as a shouting contest where decisions are made via power plays. More factual ways will look at financial returns, by considering discounted net present value calculation or return on investment. Recently there has been a lot of attention given to the notion of *Cost of Delay*.

Of all these approaches, *Cost of Delay* is probably the one that provides the most judicious prioritization criteria. However, where most such approaches fail is that they do not take into consideration the impact of the *Constraint*. As we will see, even *Cost of Delay* can be improved on.

Quantifying the Work Load

Different projects will naturally load the *Constraint* differently. Consequently, they will employ more or less *Flow Time* on the *Constraint* itself, which means that their overall *Flow Time* across the system will vary. In the simulated scenario where all projects bear the same normalized value of 100 million Euro, those that employ the least time on the *Constraint* will be the most valuable for the business, because they bring in *more money per unit of time*. What we are dealing with is the notion of *Throughput Rate* (a.k.a. *Throughput Octane* or *Flow Velocity*). If we can find ways to quantify how much money each project brings in *per unit of time*, then we can sort the projects according to that metric. In other words, we are aiming at the speed of generating profit.



Throughput Rate (TR): The rate at which the system is able to generate *Financial Throughput* (i.e. money per unit of time).

In the simulation, the coins are distributed across the ten projects and consequently, each project determines the *Work Load* for each team, and in particular for the *Constraint* team. Remember that the “work” is represented by flipping coins, and different coins have different rules as to how they are flipped and passed on. So we can count “how much work” is involved with each project. We can also assume, for further equalization of conditions, that every single coin flip takes one unit of time. Suppose one sample distribution is like the one illustrated here:

Distribution of Coins per Project

| Project | 1c | 2c | 5c | 10c | 20c | 50c |
|---------|----|----|----|-----|-----|-----|
| P01 | 4 | 3 | 1 | 0 | 1 | 1 |
| P02 | 2 | 4 | 3 | 0 | 1 | 0 |
| P03 | 5 | 1 | 2 | 2 | 0 | 0 |
| P04 | 4 | 3 | 2 | 1 | 0 | 0 |
| P05 | 5 | 2 | 3 | 0 | 0 | 0 |
| P06 | 3 | 3 | 1 | 1 | 1 | 1 |
| P07 | 2 | 1 | 2 | 3 | 1 | 1 |
| P08 | 3 | 4 | 2 | 1 | 0 | 0 |
| P09 | 0 | 5 | 2 | 1 | 0 | 2 |
| P10 | 2 | 4 | 2 | 1 | 1 | 0 |

For each project, we can now break down how this *Work Load* becomes the distinct effort (the “flipping of coins”) that hits each team, as follows:

Effort Load Breakdown (Coin Flips) per Team

| Project | A | | B | | C | | D | | | Total |
|----------------|----|----|----|-----|-----|-----|-----|-----|-----|-------|
| | 1c | 2c | 5c | 10c | 20c | 50c | 10c | 20c | 50c | |
| P01 | 4 | 3 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 12 |
| P02 | 2 | 4 | 3 | 0 | 1 | 0 | 0 | 1 | 0 | 11 |
| P03 | 5 | 1 | 2 | 2 | 0 | 0 | 2 | 0 | 0 | 12 |
| P04 | 4 | 3 | 2 | 1 | 0 | 0 | 1 | 0 | 0 | 11 |
| P05 | 5 | 2 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 10 |
| P06 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 13 |
| P07 | 2 | 1 | 2 | 3 | 1 | 1 | 3 | 1 | 1 | 15 |
| P08 | 3 | 4 | 2 | 1 | 0 | 0 | 1 | 0 | 0 | 11 |
| P09 | 0 | 5 | 2 | 1 | 0 | 2 | 1 | 0 | 2 | 13 |
| P10 | 2 | 4 | 2 | 1 | 1 | 0 | 1 | 1 | 0 | 12 |
| Total per Team | 60 | | 30 | | 10 | | 20 | | | |

Team A is the constraint because it has the largest Work Load of coins to flip.

While the above breakdown is very detailed, what becomes even more significant is the *total Work Load per team*, and in particular for the *Constraint* team. By summing up the effort components per team, we get the following table:

Aggregate Effort Load (Coin Flips) per Team

| Project | A | B | C | D | Total |
|---------|---|---|---|---|-------|
| P01 | 7 | 1 | 2 | 2 | 12 |
| P02 | 6 | 3 | 1 | 1 | 11 |
| P03 | 6 | 4 | 0 | 2 | 12 |
| P04 | 7 | 3 | 0 | 1 | 11 |
| P05 | 7 | 3 | 0 | 0 | 10 |
| P06 | 6 | 2 | 2 | 3 | 13 |
| P07 | 3 | 5 | 2 | 5 | 15 |
| P08 | 7 | 3 | 0 | 1 | 11 |
| P09 | 5 | 3 | 2 | 3 | 13 |
| P10 | 6 | 3 | 1 | 2 | 12 |

What becomes of interest, besides the total effort per project, is the effort expended at the *Constraint*. For example we see that for project P01, Team A has a *Work Load* of 7.

Economic Prioritization and Selection via *Throughput Rate*

Since all projects have the same value (100 million Euro), we can calculate:

- Each project's *Return on Investment* (ROI) - shown in the table - by dividing the value by the total *Work Load* of the project on the system.
- Each project's *Throughput Rate* (TR) - shown in the table - by dividing the value by the total *Work Load* of the project on the *Constraint* in the *Work Flow* (that we identified as described in Chapter 11 - *Finding the Constraint in PEST Environments*).

For project P01 the *Return on Investment* is 100 divided by 12 giving 8.33; and the *Throughput Rate* is 100 divided by 7 giving 14.29. The results are summarized here:

| ROI and Throughput Rate | | | | Sort by ROI | | | | Sort by Throughput Rate | | | |
|-------------------------|-------|-------|-------|-------------|-------|-------|-------|-------------------------|-------|------|-------|
| Project | Value | ROI | TR | Project | Value | ROI | TR | Project | Value | ROI | TR |
| P01 | 100 | 8.33 | 14.29 | P05 | 100 | 10.00 | 14.29 | P07 | 100 | 6.67 | 33.33 |
| P02 | 100 | 9.09 | 16.67 | P02 | 100 | 9.09 | 16.67 | P09 | 100 | 7.69 | 20.00 |
| P03 | 100 | 8.33 | 16.67 | P04 | 100 | 9.09 | 14.29 | P10 | 100 | 8.33 | 16.67 |
| P04 | 100 | 9.09 | 14.29 | P08 | 100 | 9.09 | 14.29 | P03 | 100 | 8.33 | 16.67 |
| P05 | 100 | 10.00 | 14.29 | P10 | 100 | 8.33 | 16.67 | P06 | 100 | 7.69 | 16.67 |
| P06 | 100 | 7.69 | 16.67 | P03 | 100 | 8.33 | 16.67 | P02 | 100 | 9.09 | 16.67 |
| P07 | 100 | 6.67 | 33.33 | P01 | 100 | 8.33 | 14.29 | P01 | 100 | 8.33 | 14.29 |
| P08 | 100 | 9.09 | 14.29 | P06 | 100 | 7.69 | 16.67 | P04 | 100 | 9.09 | 14.29 |
| P09 | 100 | 7.69 | 20.00 | P09 | 100 | 7.69 | 20.00 | P05 | 100 | 10 | 14.29 |
| P10 | 100 | 8.33 | 16.67 | P07 | 100 | 6.67 | 33.33 | P08 | 100 | 9.09 | 14.29 |

Normalized Throughput 76.21

Normalized Throughput 103.34

Note: When we are calculating ROI, we are using the word “investment” in the conventional sense; and not what is meant from a *Throughput Accounting* perspective - because someone calculating ROI does not know *Throughput Accounting*. In the ROI view, a coin flip is what you “invest” (or the unit cost allocation) to get work done. From a *Throughput Accounting* perspective each coin flip represents a unit of *Operating Expense*. And we know that one way to raise ROI (in the *Throughput Accounting* sense) is to lower *Operating Expense*. We are just trying here to construct an accessible *Mental Model* that can

allow us to become familiar with TA concepts from an intuitive and operational perspective.

We can then sort the table according to the calculated *Return on Investment* and then again according to the *Throughput Rate*. In terms of conventional *Return on Investment*, the best results are achieved by delivering (in order) projects P05, P02, P04, P08 and P10. In terms of *Throughput Rate*, the best results are achieved by delivering (in order) projects P07, P09, P10, P03 and P06.

Note: You might notice that projects P06 and P02 have the same *Throughput Rate* of 16.67. In sorting by TR, they are in the fifth and sixth position; but because the exercise asks us to pick only the five most significant projects, one of the two has to be left out. Since they both have the same TR, it wouldn't matter which one we picked (unless, of course, there are other tie-breaker reasons known to the product owners and/or stakeholders - in which case we would resolve it within the *Ranking* phase of the *Full-Kitting* activity as explained in *Chapter 18 - Full-Kitting as Ongoing Executive Activity*). Looking at the table, we might be tempted to pick project P02 rather than project P06 because we see that the former has a higher ROI (9.09) than the latter (7.69). However, the ROI we are seeing in the table is the ROI calculated on a *Cost Accounting* basis, it is not the ROI calculated according to *Throughput Accounting*. If we were to calculate the ROI according to the equations of *Throughput Accounting*, both projects would produce the *same* ROI precisely because they have the same *Throughput Rate*. So picking project P02 instead of project P06 would not make a difference. And we picked project P02 precisely to illustrate this subtle difference so that we don't fall into the *Cost Accounting* trap!

There is something remarkable going on here.

Of all the projects selected by the ROI criteria, only project P10 qualified on the top five list under a *Throughput Rate* analysis. Undertaking projects P05, P02 ,P04, and P08 would be suboptimal and lead to the inevitable inferior financial performance that was outlined in *Chapter 7 - Accounting F(r)iction*.

We can then compare the two approaches with respect to the *Throughput* they will produce. The *Return on Investment* option comes in at 76.21, while the *Throughput Rate* option delivers more value at 103.34.

Here we are *not* comparing *Return on Investment* to *Throughput Rate* directly. It cannot be done because they are dimensionally incongruous. Furthermore ROIs are not additive. The rationale is: we are looking at the *Throughput* of a project ranked on the basis of *Return on Investment*, and then we compare it to the *Throughput* of the other project that has the same rank, but established on the basis of *Throughput Rate*, which is additive within the same constrained system.

Since we have to select the five most profitable projects, we follow the same logic for the topmost five ranked projects according to the two criteria. Finally we compare the aggregate *Throughput* of the five projects, according to the two ranking criteria.

At the bottom of the table we add up what we call *Normalized Throughput*. What does it mean? It is "normalized" in the sense that it is the *Throughput* that the five projects would jointly generate per unit of time.

And it corresponds directly to the financial performance of the system.

Note: The *Throughput Rate* that we have used in our financial analysis is the amount of money generated at the *Constraint* per unit of time. This is an additive ratio. Understanding that it is additive is very important. Understanding that it is additive with respect to the *same Constraint* is even more important. Let's try to understand this with an analogy. Let's consider speeding cars. If we travel with a car on one road (the *Constraint*) at 50 mph for one hour, and then change cars and travel *on the same road* at 100 mph for another hour, at the end of the two hours the cumulative distance travelled will be 150 miles - *on that very same road!* If we do the same, but with two cars that travel on *different roads*, then the distances travelled are not additive: the distance travelled by the second car on the second road does not move us an inch further on the first road.

Another way to understand this comparison is to imagine that we run our system for one time unit working on the first project, then a second time unit on the second project: and so on, until the fifth time unit on the fifth project. In principle, after five time units we will have generated a *Throughput* equal to the sum of the *Throughput* (per time unit, of course) of the five individual projects.

The difference is in how we select these five projects.

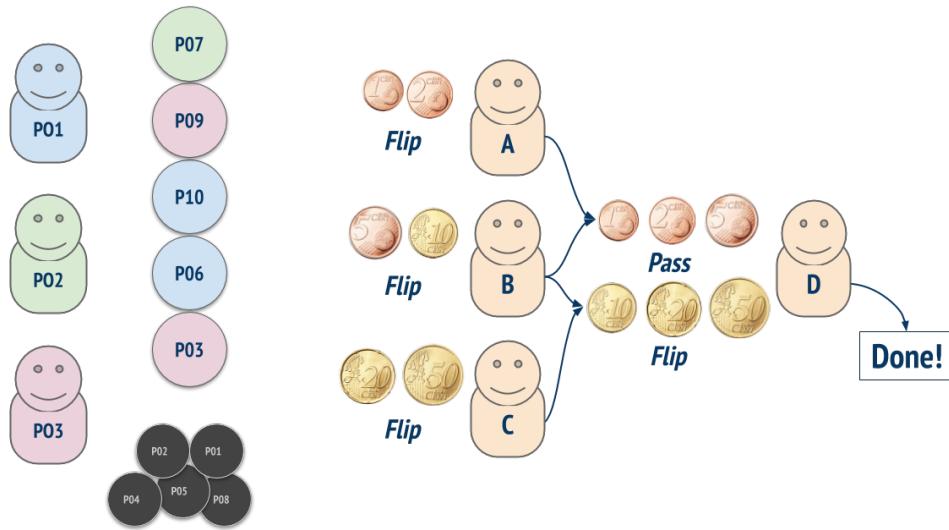
In the first case, we pick the topmost five projects based on ranking according to *Return on Investment* generated through the effort expended by all teams. In the second case, it is based on ranking according to the *Throughput Rate* on the *Constraint* team alone. The first case generates a *Normalized Throughput* of 76.21, while the second case generates a *Normalized Throughput* of 103.34.

Clearly, the selection and sequence determined by means of *Throughput Rate* are far more superior than a simple *Return on Investment* calculation. In this example, *Throughput Rate* ranking will deliver over 35% (that is the percentage increase from 76.21 to 103.34) more *Financial Throughput* than the *Return on Investment*.

Any Chief Financial Officer - or manager with Profit and Loss responsibilities - who ignores the benefits of *Throughput Rate* is losing money for their companies. They should learn to run their operations according to the priorities established through the practices of *Throughput Accounting*.

Note: At the end of the journey, traditional ROI and Net Profit have a clear purpose for measuring profitability with "accurate" ex-post (after the fact) information. They will always reflect the real historical outcome. But in selecting projects, ROI - and for that matter IRR, NPV etc. - is an inferior ex-ante (forecasting) tool because by ignoring the *Constraints*, it is impossible to know what the real speed of profit generation is. This is why *Throughput Accounting*'s use of "relevant" information at the *Constraint* will always prevail over ROI for selection of the most profitable projects. At the economic level, including the cost of *Non-Constraint* resources is just wrong.

Five projects have been excluded by this selection. After using the *Throughput Rate* criteria to effect the prioritization and selection, the project list looks as follows, where the excluded projects are appear in black:



The discarded projects will be there for the prioritization and selection for the next business period or deadline, if the exercise of these options still has worth.

Cost of Delay and CD3

Any discourse about prioritization and sequencing would not be complete without considering *Cost of Delay* and *CD3*.



Cost of Delay (CoD): A measurement of urgency expressed in terms of loss of value over time. Typically the value is denoted in money; but it can also be described with other quantitative parameters or even in qualitative / subjective terms. The intent of CoD is to give a sense of how the value decays as time passes. It can be considered as the cost of allowing that period of time to pass without realizing the value.

Note: *Cost of Delay* is a ratio of *money over time*. What immediately gets our attention is that this ratio is dimensionally congruent to the ratio that defines *Throughput Rate*, which is also money divided by time.



CD3: An acronym that stands for “*Cost of Delay Divided by Duration*.” It represents the “rate of change” of *Cost of Delay* - in other words, the change in speed of the loss of value over time. It is used for prioritizing between different options on the basis of their *Cost of Delay* in relation to the time needed for their respective implementation.

The objective of *CD3* is to prioritize in order to maximize the value delivered per unit of time, by allowing us to compare options with different *Cost of Delay* and different implementation durations. This intent is similar to what we achieve with *Throughput Rate*. Dividing by the implementation duration makes it possible to compare projects directly, notwithstanding that their implementations take different amounts of time. This is also similar to what happens with *Throughput Rate*, where the division is by the duration of the project on the *Constraint*, rather than its duration throughout the entire system. Clearly there are enough similarities between *CD3* and *Throughput Rate* to warrant the question: Which makes more sense from a business perspective?

In order to have some common ground here, we need to relate the notion of *Cost of Delay* and *CD3* to the setting of our simulation. Remember: the simulation already involves a number of simplifying assumptions, in order to have a comprehensible *Mental Model*. In the simulation we assume that:

- The action of flipping a coin is considered a *unit of effort*.
- The action of flipping a coin takes one *unit of time*.
- All projects have a normalized value of 100 million Euro.

These assumptions are particularly important when considering the *Throughput Rate* in the real world. Because the *Throughput Rate* is defined as the amount of money that the system produces per unit of time, when operating “for real” we would be concerned about the time forecasts/estimates on the *Constraint* to be used as the denominator for *Throughput Rate* calculations.

This is what allows us to bridge our *Mental Model* to the *Cost of Delay* or *CD3* ratios - because these ratios also present a duration (i.e. time) as their denominator.

Can we derive something that is representative of *CD3* in our simulation? The fact that all our projects are worth the same (100 million Euro each) is significant. It means that they all have the same *Cost of Delay*. To appreciate why, imagine that the 100 million Euro were all collected as cash the very same day a project is delivered. Of course this is yet another simplification supporting our *Mental Model*; but it is a powerful one, because it allows us to state that for every day of delayed delivery, we are postponing the collection of 100 million Euro.

We now need to get to the *CD3* which allows for meaningful comparisons between projects that have *different durations*. We have the project *Cost of Delay* (always normalized to 100 million Euro for every project) - so we only need to divide by what would be the *duration* of the projects from *start to finish* to get the *CD3*.

Can we find out what the duration of each project is?

Notice that given the topology of the *Work Flow*, we have a degree of parallelism between Team A, Team B and Team C. They are all working concurrently before handing over their deliverables to Team D. We can expect that the duration of a project is the maximum time taken by teams A, B and C, plus the time taken by Team D. In tallying this, we must not forget that only the coins that are actually flipped will consume their unit of time, according to the coin flipping rules that we already know.

Let us revisit the table with the *Aggregate Effort Load per Team*, illustrated again below, but this time interpreting the numbers as units of time, rather than units of effort (because they are equivalent in virtue of our simplifying assumptions):

Aggregate Effort Load (Coin Flips) per Team

| Project | A | B | C | D | Total |
|---------|---|---|---|---|-------|
| P01 | 7 | 1 | 2 | 2 | 12 |
| P02 | 6 | 3 | 1 | 1 | 11 |
| P03 | 6 | 4 | 0 | 2 | 12 |
| P04 | 7 | 3 | 0 | 1 | 11 |
| P05 | 7 | 3 | 0 | 0 | 10 |
| P06 | 6 | 2 | 2 | 3 | 13 |
| P07 | 3 | 5 | 2 | 5 | 15 |
| P08 | 7 | 3 | 0 | 1 | 11 |
| P09 | 5 | 3 | 2 | 3 | 13 |
| P10 | 6 | 3 | 1 | 2 | 12 |

We can see that for project P01, Team A will consume the longest time (7 units) compared to Team B (1 unit) and Team C (2 units). Then Team D will do its part consuming 2 units. So we can assume that project P01 will take 9 units to complete.

Another example could be project P07, where Team B will take the longest time (5 units) compared to Team A (3 units) and Team C (2 units). Then Team D will consume 5 units, and in total project P07 will take 10 units to complete

Note: Observe that for project P07 the longest time is taken by a *Non-Constraint* team - and this is due to the variability that is found in the *Work Load*. This particular instance will not change the fact that in the statistical aggregate of the entire *Work Load*, Team A remains the *Constraint* (in relation to this particular distribution of coins, of course).

We can repeat the calculation for all projects, and see the duration they take to flow through the system. (Another simplifying assumption is that we are not *Multitasking*, so one project will be executed after another, with no overlap).

The table looks like this:

Aggregate Project Duration

| Project | A | B | C | D | Duration |
|---------|---|---|---|---|----------|
| P01 | 7 | 1 | 2 | 2 | 9 |
| P02 | 6 | 3 | 1 | 1 | 7 |
| P03 | 6 | 4 | 0 | 2 | 8 |
| P04 | 7 | 3 | 0 | 1 | 8 |
| P05 | 7 | 3 | 0 | 0 | 7 |
| P06 | 6 | 2 | 2 | 3 | 9 |
| P07 | 3 | 5 | 2 | 5 | 10 |
| P08 | 7 | 3 | 0 | 1 | 8 |
| P09 | 5 | 3 | 2 | 3 | 8 |
| P10 | 6 | 3 | 1 | 2 | 8 |

Note: As we learned in *Chapter 5 - Where to Focus Improvement Efforts*, it is not always the overall shortest job that is the one that brings the most value. In the table above we see that project P07 has the *longest* duration of all, but as we can see (in the table below) it is also the project that brings in the *highest* value - the *Throughput Rate* is 33.33, and all this because it engages Team A - the *Constraint* team - for only 3 time units: the least of all projects.

Finally! Knowing that every project has a *Cost of Delay* of 100 million Euro per unit of time of delay, we can calculate the CD3 by dividing by the duration. Then, as we did before, we can select the five topmost projects. We can derive their normalized *Throughput Rate* and thus compare the CD3 prioritization to the *ROI* and the *Throughput Rate* prioritization.

We arrive at this result:

| CD3 and Throughput Rate | | | | Sort by CD3 | | | | Sort by Throughput Rate | | | |
|-------------------------|-------|-------|-------|-------------|-------|-------|-------|-------------------------|-------|-------|-------|
| Project | Value | CD3 | TR | Project | Value | CD3 | TR | Project | Value | CD3 | TR |
| P01 | 100 | 11.11 | 14.29 | P02 | 100 | 14.29 | 16.67 | P07 | 100 | 10.00 | 33.33 |
| P02 | 100 | 14.29 | 16.67 | P05 | 100 | 14.29 | 14.29 | P09 | 100 | 12.50 | 20.00 |
| P03 | 100 | 12.50 | 16.67 | P09 | 100 | 12.50 | 20.00 | P10 | 100 | 12.50 | 16.67 |
| P04 | 100 | 12.50 | 14.29 | P03 | 100 | 12.50 | 16.67 | P03 | 100 | 12.50 | 16.67 |
| P05 | 100 | 14.29 | 14.29 | P10 | 100 | 12.50 | 16.67 | P06 | 100 | 11.11 | 16.67 |
| P06 | 100 | 11.11 | 16.67 | P04 | 100 | 12.50 | 14.29 | P02 | 100 | 14.29 | 16.67 |
| P07 | 100 | 10.00 | 33.33 | P08 | 100 | 12.50 | 14.29 | P01 | 100 | 11.11 | 14.29 |
| P08 | 100 | 12.50 | 14.29 | P06 | 100 | 11.11 | 16.67 | P04 | 100 | 12.50 | 14.29 |
| P09 | 100 | 12.50 | 20.00 | P01 | 100 | 11.11 | 14.29 | P05 | 100 | 14.29 | 14.29 |
| P10 | 100 | 12.50 | 16.67 | P07 | 100 | 10.00 | 33.33 | P08 | 100 | 12.50 | 14.29 |

Normalized Throughput 84.29 Normalized Throughput 103.34

Seemingly, the *CD3* prioritization has merit, because it produces a *Normalized Throughput* of 84.29, which is certainly better than 76.21 produced by the *Return On Investment* ranking.

Yet it is a far cry from 103.34 produced by *Throughput Rate* ranking.

In this case using *Throughput Rate* for the ranking will still deliver over 22% (the percentage increase from 84.29 to 103.34) more *Financial Throughput* than the *CD3* alternative.

Note: What is striking here is that *CD3* - like *ROI* - will rank last project P07, which has the *highest Throughput Rate*!

Now, there is a caveat. The scenario examined in the tables above, is actually a best case for *CD3* (again, with respect to this particular *Work Load* distribution). Let's see what it is about.

There are five projects that all have the same *CD3* of 12.50. The five projects are: P09, P03, P10, P04 and P08. Only three of them made it into the top-5 list. Not coincidentally, the three projects that had the best *Throughput Rate*. Of course, this particular choice was intentional here, simply because we know the *Throughput Rate* and we want to see how well the *CD3* prioritization can actually fare.

In the real world, all five of those projects would appear to be of the same importance (from a *CD3* perspective). The metric of *Throughput Rate* would simply not be known, let alone considered. Projects P04 and P08 that did not make it into the top five, could have been selected equivalently, in place of projects P09 and P03. From a *CD3* viewpoint, nothing would change.

However, considering the corresponding *Throughput Rates* which we now know about, we see that their *Throughput Rate* is respectively 14.29 and 14.29 (P04 & P08) instead of 20.00 and 16.67 (P09 & P03). The resulting normalized *Throughput Rate* would be 76.21 - which coincidentally turns out to be

the same as the one produced by the *Return on Investment* ranking.

In this particular case, with this specific *Work Load* distribution, and this precise choice of projects, the *CD3* prioritization would not be any better than the *ROI* prioritization. In fact, it is even conceivable that in certain edge cases, *CD3* could be even worse than *ROI*! The devil is in the details.

Note: Also consider that the same kind of situation could arise when using *ROI*: projects with different *TR* could still have the same *ROI*. The selection based on *ROI* would not detect a difference, but the impact in terms of *TR* could be very significant.

Further Notes on *Cost of Delay* and *CD3*

From the above exercise, we must acknowledge that even though *CD3* is all the rage - especially with SAFe practitioners - if we ignore the *Constraint*, we are basically leaving money on the table.

SAFe practitioners attribute job size as a proxy variable to duration when calculating *CD3*. In knowledge-work, there is no correlation between job size and duration, particularly in low *Flow Efficiency* environments.

Note: In the simulation, we can get away with the equivalence, because it is a simplifying assumption to gain understanding. While that heuristic can guide our choices in the real world, we must not believe that our simplifying assumptions still hold there.

If time is money, then the denominator of SAFe's *Cost of Delay* is simply not quite up to the challenge. If one wants a robust time denominator for knowledge-work, then the *Theory of Constraint*'s focus on how fast jobs go through the *Constraint* of the *Work Process* - the *Throughput Rate* - is a much more sound and scientific approach. It correctly captures the context where we have "pure" time as a mathematical denominator in a value/time ratio.

Note: There might be other characterizations of *Cost of Delay* popular with the *Kanban Method*, which relate to *Delay Cost Profiles* connected to *Class of Service*, and which have a more qualitative / subjective assessment of the *Cost of Delay*, because the value component is not directly quantifiable. Examples of such profiles include: *Expedite*, *Fixed Date*, and *Intangible*. We won't explain these here. However, while we are definitely more concerned about the quantitative approach based on *CD3*, these qualitative and subjective assessments of *Cost of Delay* do play a role in ranking resolution during *Full-Kitting* activities. More details about this kind of usage are given in *Chapter 18 - Full-Kitting as Ongoing Executive Activity*.

The concept of *Cost of Delay* has gained a lot of traction in recent years - possibly because of the

ingenious idea of naming the concept with the word “*cost*.” It thus becomes a concept that all the *Cost Accounting* practitioners can blissfully relate to - it is a “*cost*!”

It is unfortunate that “*Throughput Rate*” does not have a similarly seductive name for the accountants; and this very fact makes it less accessible.

However, the notion of *Cost of Delay* is bringing a very important element to the purview of the *Cost Accounting* world: a more reasonable understanding of the significance of the *time* dimension. But *Cost of Delay* still fails because it ignores the presence and the impact of the Constraint.

The Need for Estimation Practices

The example used in the simulation is fictitious. The intent is always that of making us *think* so that we can have a sharp *Mental Model* to understand what the trade-offs of various decisions are, and hopefully gain some deeper insights that will be of practical guidance on a day-to-day basis.

One obvious contrivance was the simplified view that all projects had the *same* market value (of 100 million Euro). The purpose was to force us to think about how our decisions about prioritizing and selecting (apparently equivalent projects) can have very significant differences in terms of the economic benefits that the business can reap. The skills we learned through this exercise are applicable even to real scenarios, where the different projects we undertake will undoubtedly have different values.

That being said, having the knowledge that different projects will engage the *Constraint* in different ways will induce us to pay more attention to how we prioritize and select the projects that we want to deploy into the *Work Flow*. The higher the *Throughput Rate* on the *Constraint* is, the more money the company will make per unit of time. Knowing this will give us great guidance.

In the simulation we could also contrive a simple way to have some measurement of the *effort* that each project needed. The “unit of effort” was simply the flipping of a coin and the underlying assumption was that every flipping of a coin would consume the same amount of “resources” - and consequently every flip also had a normalized “duration.” Being the equal time (in principle) for every coin flip, the simulation did not have to take into account how much time the “unit of effort” actually consumed. It is as if every unit of effort was performed in the same unit of time.

In reality, the *time* dimension is one of the most important elements we need to consider in running our businesses.

As we get more familiar with the notion of *Throughput Rate* we will think about it more in terms of *Financial Throughput* rather than in terms of *Operational Throughput*.

In the exercise, *Throughput Rate* was introduced as the ratio between the monetary value of each project and its *Work Load* on the *Constraint*. In the real world, because of the importance of *time-value-for-money*, we need to grasp that *Financial Throughput Rate* is the amount of money produced by the system per unit of time.

Therefore, the denominator is more aptly expressed in time, rather than as *Work Load* (even though *Work Load* can be a proxy of time). Again, we are using a *Mental Model* to gain understanding.

Prioritization and sequencing of projects is done *before* the projects are actually released into the *Work Flow*. Unfortunately, given the timeline, it is also the moment in the project’s life cycle when the least is known about it.

Yet if we want to perform prioritization and selection on the basis of economic benefits the *denominator must receive a value*.

Estimation/Forecast of duration, not of due dates

Being able to estimate how much time a project will take becomes critical in order to be able to apply this approach; because the prioritization/selection criteria will be based on how much money every project can produce per *unit of time* on the *Constraint*. Therefore, we need to estimate the *Flow Time*. An estimation of how long the project will keep the system busy - and in particular what the *Flow Time* on the *Constraint* will be.

We are *not* estimating *when* any given project will be delivered. We focus only on its duration: *how long* it will take the system to deliver. Since we are scheduling work according to the pace of the *Constraint* (the “drum” in the *DBR*), and we want to ensure that the project will flow *uninterruptedly* through the system, we can expect to gain a fair estimate - especially if we keep in mind what we learned in the first two chapters about managing our system so that it is stable and predictable.

The next question is how to perform such an estimation.

While in the *TameFlow Approach* there is *no* prescribed manner of making such estimations - as long as we are comfortable and have confidence in their outputs - we will suggest two:

- T-Shirt Sizing (or Reference Class Estimation)
- Probabilistic Forecasting with *Flow Metrics*

We will not go into details - not even explain how these estimation/forecasting methods work - since there are other references that can do this much better.

T-Shirt Sizing

We do not want to burden the engineering/knowledge-worker teams with lots of work for estimating what is to come. In fact, the less we engage the teams, the better. Their skills are better employed at getting work done, rather than estimating how long it will take. *T-Shirt Sizing*, if done properly, has the advantage of using very little engineering time, and yet arrive at reasonable estimates.

Another aspect to consider is that the engineers’ expert opinions is incorporated into the estimates. It lets them have conversations about the work to come and develop a common understanding/vision of upcoming challenges. This has a subtle psychological effect of creating more engagement and ownership.

T-Shirt Sizing is very social and consensus-driven. On the down side it can still be very expensive, both in terms of costs and potentially in terms of psychological drain on the participants - notwithstanding that, from the authors’ viewpoint, it generally delivers better results than other expert-opinion based estimation practices. The preferred technique is to use probabilistic forecasting, as we will see next, because it requires no engineering time at all to arrive at reliable forecasts by simply leveraging your empirical flow metrics.

Probabilistic Forecasting with *Flow Metrics*

If we are already diligently collecting *Flow Metrics*, we can use *Little’s Law* to get a probabilistic forecast of the duration of new projects. Of course, in order to do that, we must ensure that the conditions of applicability of *Little’s Law* are present, as explained by Daniel Vacanti in his *Actionable Agile Metrics for Predictability* book.

Note: To know more about how to perform probabilistic forecasting and about *Little's Law*, see Daniel Vacanti's work just mentioned above. His approach is entirely compatible with and recommended by the *TameFlow Approach*. In particular you can also read the review of his book in the *TameFlow Chronicles 2011-2015* (which you can find in the [free bonus downloads](#)) to see how his techniques fit into the *TameFlow Approach*.

The main advantage of this method is that it can be automated, and does not require *any* engineering effort at all - except for an initial acknowledgement that a requirement is clear enough and feasible.

Advanced Application: Estimate the *Flow Time* on the *Constraint* alone

The prioritization and selection technique we learned about in this chapter is based on ranking according to *Throughput Rate*. We also learned that we can calculate the rate by using the *Flow Time* on the *Constraint* as the denominator of the ratio.

Now, recall how in the *Prologue* we hinted at "*The Extra Step 6*" whereby we make a deliberate effort to keep the *Constraint* in a known place? Well, if we are in such circumstances that our incoming *Work Load* has a relatively known statistical distribution (the "shape and form"), or we can strive to perform aggressive *Demand Shaping* so that it does, then we have the opportunity to drastically simplify our estimation/forecasting efforts.

In the PEST scenario of the simulation - where Team A is the *Constraint* relative to the incoming *Work Load* - we need to have estimates *only for Team A*. The other three teams need not bother producing any estimates at all.

This saves both time and effort.

If the shape and form of the incoming *Work Load* changes so that a different team becomes the *Constraint* in the *Work Flow*, then the only estimates that are needed are those of *that* team; but we want to avoid this precisely with the policy of deliberately keeping the *Constraint* in a well known place.

This means that all teams need to be instrumented and ready to provide estimates/forecasts whenever they happen to become the *Constraint* in the *Work Flow*. This is similar to how the teams need to be ready to activate their *Throughput Board* in the same situation to be able to manage the *Constraint* in *their Work Process* (as we saw in earlier chapters).

TameFlow teams are always prepared. Like Boy Scouts. Because they know that at any moment they could become the next Herbie.

When operating in this mode, the *Full-Kitting* activity (see *Chapter 17 - Introduction to Full-Kitting* and *Chapter 18 - Full-Kitting as Ongoing Executive Activity*) becomes even more important. As we will see, *Full-Kitting* entails estimating/forecasting how long each project will be. This is necessary in order to build up the virtual *Work Load* queues that will tell us where the *Constraint* in the *Work Flow* is. Then we use the estimates/forecasts for that *Constraint* team to perform the *Throughput Rate* prioritization. Now if we know where the *Constraint* is located because we are able to keep the whole system stable enough through active *Demand Shaping*, we do not need to check the queues in front of all other teams.

However, we need to be alert to changing circumstances, that escape our effort to keep the system stable, so that we can nonetheless detect that the *Constraint* is moving to some other place. This is why during the *Full-Kitting* activity we must always perform a thorough analysis of the recent past performance of every team. As long as the system is stable, the *Constraint* team will present the lowest *Throughput* metric (in terms of projects executed), while the other teams will be more performing.

If the gap in performance closes, we must decide if that was a contingent situation (the “things that happen” during the *Journey*) and thus assume the gap is temporary and passing, and do not do anything special about it; or if it is due to the *Work Load* changing its shape and form (we are entering “another” *Jungle*).

In this latter case we should again go back to normal operations and observe how the new *Work Load* changes the queues in front of the teams, and detect which team is the new *Constraint* in the *Work Flow*.

Naturally for all this to work, the entire organization must be highly attuned and skilled at applying the *TameFlow Approach*.

In nostalgic retrospect of the early days of *eXtreme Programming* (XP) this way of approaching estimates/forecasts might be the epitome of the Agile maxims: “*You Ain’t Gonna Need It*” (YAGNI) and Agile Principle of “*Maximize the amount of work not done.*” The best estimates/forecasts are those that are not needed.

We need estimates/forecasts only on the *Constraint* in the *Work Flow*. Imagine the impact on organizations with 10s or 100s of teams where the *Tameflow Approach* was successful in smoothing the entire value stream of non value and time consuming activities. An ode to *Management by Exception* that is the essence of the *TameFlow Approach*, and completely lacking in Agile, Kanban and all other improvement models.

Takeaways

Sequencing and prioritizing in knowledge-work is perishable. Within a day or even an hour, our notion of business value can change radically. Examples: a new announcement from a competitor; a supplier going bankrupt; a new regulatory requirement. No matter what might be the latest insight, speed through our value stream cannot be dismissed. It is under our managerial control and not whimsical to the changing business imperatives of the day.

And this is the key.

Using the *Throughput Rate* at the *Constraint* eases our pain and we can simply switch the sequencing/prioritizing quagmire into this: pick what we want, it will be done as fast as can be. No need for sophisticated math any more.

This is another *Mental Model* that can be replaced by automation alone. The human brain is weak at integrating complex issues such as sequencing and prioritizing with math. Automation can give us guidance, but ultimately we will have to use our insight and intuition. More about this in *Chapter 18 - Full-Kitting as Ongoing Executive Activity*.

Emiliano Sutil says...

I am always looking for new ideas that can help me in the way I manage my products, projects, teams, process. I can safely say that this book - “Tame your Work Flow” - is one of those books that have the characteristics of what I consider as the original Agile Revolution: Question preconceived ideas, break with old paradigms, represents new challenges and mental models... in short, it is a true revolution. It has that spark!

I am a big fan of the Kanban Method; yet the book made me question some elements that I took for granted such as limiting the WIP by columns. I look for books that challenge the status quo and this book has succeeded.

The examples, metaphors, simulations, etc. are brilliant and help the reader to have a deeper understanding of another of the elements that I am most passionate about: The Theory of Constraints. It describes it in a way that makes you look at it with other eyes. Another important element of the book is how it handles what is a harsh reality for many of us: The “PEST” environments. Thank you for giving us a help with that! Finally, the set of patterns described at the end are a very useful tool to apply the ideas presented in the book in a practical way.

In short, this book is a must read for anyone who wants to improve the way they deliver value to their customers.

Emiliano Sutil, Project Manager, Scrum Master, Agile Coach, DevOps Expert at XERIDIA

PART 5—Preparing for High Performance Execution and Governance

In a PEST environment there are many things going on, all at the same time. Conventional management practices are very much *top-down*. Conventional Agile practices are typically *bottom-up*. The *TameFlow Approach* is both *top-down* and *bottom-up* and the glue that keeps it all together is *Informational Flow*.

Information needs to be created and made available at the right time to the right people.

For this to happen the nervous system of the organization needs rewiring. In particular, the nerve endings detecting environmental changes and the synapses transmitting information from one part to another of the organization need to be founded and established.

Because the action happens on the ground floor, establishment of this nervous system starts from there, and goes upward to the top management echelons. At all levels, no matter the pay grade or rank, all people need to have a shared understanding of what is going on. They all need to *interpret* the signals and the information carried by the new organizational nervous system in a coherent, consistent and non-contradictory way. That's where the shared *Mental Models* come into play with all their power, combined with the patterns of *Unity of Purpose* and *Community of Trust*.

We are talking about decentralised decision making. Where everybody on the floor, at all levels of the organization, is empowered to quickly make decisions and take charge of actions, without layers of approval levels. (See the *Essence of TameFlow* booklet for how shared *Mental Models* work in this direction).

In this Part 5 we will examine the components and elements that make up the new organizational nervous system. In Part 6 we will discover the full impact and extent of what this new nervous system enables.

Wolfram Müller says...

As a co-author of the first book of the TameFlow-series, I really had to have a close look at what Steve and Daniel were doing here!

This is not a book – this is a firework of breakthrough ideas! I tried counting them, and found on average two worthy ideas per page. For every page of the whole one book. Amazing!

So, for all managers in the world, this is a must read!

Wolfram Müller, Founder of BlueDolphin – a community platform for self-organized change based on the Theory of Constraints; co-author (with Steve Tendon) of “*Hyper-productive Knowledge-Work Performance, The TameFlow Approach and its Applications to Scrum and Kanban.*”

14—Flow Efficiency, DBR and TameFlow Kanban Boards

In the preceding chapter we learned how to zoom in on the team that is the *Constraint in the Work Flow*. The *Constraint* team in the *Work Flow* can be identified by observing which team has the highest work load. It also became clear that new work should be released into the *Work Flow* by keeping pace with the rate at which the *Constraint* (team) can pull in new work.

Since this team is limiting the *Throughput* of the entire system, we need to apply the *Five Focusing Steps*:

- **Step 1 - Identify the Constraint:** performed by looking at the *Work Load* hitting the teams and identifying the one with the heaviest queue or *Work Load* piling up in front of it.
- **Step 2 - Exploit the Constraint:** done by introducing the *Buffer* in front of the *Constraint* in order to ensure that it never runs out of work and does not waste its time. Time lost on the *Constraint* is money lost forever by the entire system.
- **Step 3 - Subordinate to the Constraint:** achieved via *DBR Scheduling*. All other *non-Constraint* teams are subordinating to the *Constraint*, making sure that they all work at the same pace and are synchronized to the capacity of the *Constraint* in order to avoid producing unnecessary WIP in the system.
- **Step 4 - Elevate the Constraint:** this is where the action required is not self-evident because it will be specific to the particular situation. Typically this is when we need to change the *Work Process* so that the *Touch Time* on the *Constraint* becomes shorter.
- **Step 5 - Repeat:** “Lather, Rinse, and Repeat!” Go back to Step 1.

Note: In situations where there is no good *Flow* and it is difficult to identify the *Constraint*, one way to start is to just pick something as if it were the *Constraint*. The resulting clearer priorities - even if not optimal - will help smooth out the *Flow* and possibly point out the real *Constraint*.

We can possibly increase the capacity of the *Constraint* team in two ways:

1. Relocate some of its *Work Load* to the other teams that have less work to do. Naturally, this requires fungibility so that these other teams are equipped (with the right skills, knowledge, capabilities, etc.) to perform that kind of work.
2. Break some silo-based territoriality, because often other teams are on different reporting lines within the company hierarchy. Both of these limitations can be addressed by employing further elements of the *TameFlow Approach*.

Notice one important aspect of the two elevation strategies just mentioned: they seek assistance *outside* of the *Constraint* team itself. In other words, from the perspective of that team, it is necessary to reach out to the *Sphere of Influence* of the people involved.

So one key question remains: can the *Constraint* team do anything directly on its own behalf to increase its capacity? Can the elevation be done by remaining (within reason) inside the process's/team's own *Span of Control*?

If we consider that the *Constraint* team is most likely performing a number of interrelated steps when executing their work, we can rationalize according to the principles of *Constraints Management*. This is where we need to *find the Constraint inside the Work Process* - the *Work Process* used by the team itself - and then manage it accordingly with *DBR scheduling*.

If the *Constraint* team is working according to a sequence of interrelated steps, let us assume that the team is already familiar with - and is indeed using - an appropriate *Kanban Board* to visualize and manage their work.

New Types of *Kanban Boards*

Unfortunately, mainstream *Kanban Boards* designs are not helpful to detect - let alone manage - the *Constraint* in the *Work Process*. We have already mentioned in the previous chapter that *Column WIP Limits* have more negative effects than positive ones.

There are further reflections regarding *Column WIP Limits* in conventional *Kanban Boards*. It is a structural weakness and part of the method's history.

One reason *Kanban Boards* are where they are with *Column WIP Limits*, is that the model was never designed to deal with *networks* of processes and queues. Probably this was a consequence that the method was originally devised around the activities of a single team and then just a few teams, with a linear and sequential *Work Flow* connecting the activities from one team to the next.

There never was a need to model *networks* of processes and activities, and to model the virtual queues *in front* of the teams inside such networks. Single boards - even if used in arrays - were an incomplete model.

As adoption became more widespread, hierarchies of boards started to become the means to manage the complexity. Klaus Leopold's *Flight Levels* are an example of such attempts. But just cobbling together *Kanban Boards* - even with hierarchies - is not adequate to represent the ever-changing dynamic interactions between the dimensions of *Work Load*, *Work Flow*, *Work Processes*, and *Work Execution*.

Some practitioners, like the brilliant Frode Odegard with his *Lean Systems Framework*, were able to resolve the *Kanban Method's* workflow linearity problems regarding what was being transmitted and what was being built at the architecture level. A workflow is not just a driver, it is a *consequence* of what happens in these two aforementioned architectural dimensions. In any sizeable organizations, the *Work Flow* is seldom linear.

In the rest of this chapter we will examine how we can evolve from a conventional *Kanban Board* design to support visual management of high-performing organisations. In particular, we will discover board designs that can be arranged in complex networked configurations to better manage *Work Load*, *Work Flow*, *Work Processes* and *Work Execution*.

The three new *Kanban Boards* can all be derived from the conventional *Kanban Boards*. These *Kanban Boards* also build on top of one another, so that we can have a choice as to how much instrumentation must be activated or deactivated as circumstances dictate.

The new boards are described below.

In Part 1 of this book, we learned about the benefits of improving *Flow Efficiency*. Most *Kanban Boards* do not help in collecting this simple metric. Often one would be forced to derive it through calculations based on unvalidated assumptions. So the first transformation we will seek to achieve is how to design a *Kanban Board* in order to truly *measure, manage* and *improve Flow Efficiency*.



Flow Efficiency Board: A *Kanban Board* where all columns are divided in two semi-columns - *Waiting for...* and ...*In Process* - in order to faithfully capture the *Wait Time* and *Touch Time* components of the *Flow Time*. The board allows for: (1) the correct computation of *Flow Efficiency*; and (2) the identification of the *Constraint* in the *Work Process*, via the longest average in-state (i.e. column) *Flow Time*.

What is even more important is that this design will allow us to perform Step 1 (Identify the *Constraint*) of the Five Focusing Steps (5FS) with respect to the process visualized by the new *Kanban Board*. It is the column that has the longest average *Flow Time*. This is something that conventional *Kanban Boards* do not allow you to seize (no wonder that Dr. Goldratt's deep insights get lost when using mainstream *Kanban Boards*!).

In Part 2 of this book we learned how important it was to focus any improvement initiative exclusively at the *Constraint*. All other kinds of improvements will produce negligible effects and create permanent costs and risks. While the *Flow Efficiency* board design allows us to identify the *Constraint* in the *Work Process*, we still need the means to perform Step 2 (Exploit the *Constraint*) of the 5FS, so that we can (ideally) ensure that the *Constraint* will never starve. We also need to perform Step 3 (Subordinate to the *Constraint*) so that all *Non-Constraint* steps do not overproduce and keep pace with the pace set by the *Constraint*. For this purpose we introduce the *Drum-Buffer-Rope Board*.



Drum-Buffer-Rope (DBR) Board: A board where the *Waiting for...* semi-column of the *Constraint* in the *Work Process* also acts as a *Buffer* to support *Drum-Buffer-Rope* scheduling for the release of the *Work Load* into the *Work Flow* and/or *Work Process*.

Another extremely important function that is achieved with a *DBR Board* is that it allows us to remove all the *Column WIP Limits* to better support unhindered *Flow* and doing a better job at reducing *WIP* than the actual using *Column WIP Limits*. We thus avoid the artificial limitation on the maximum flow *Capacity* of the system, and all the turbulence and additional variability that emerges from the unnatural limitations of *Column WIP Limits*. We are moving away from the *Kanban Method*'s pull and toward *Tameflow's Flow!*

In subordinating to the *Constraint*, it is also important to quickly react to problems that can arise in the *Work Process* surrounding the *Constraint*. It is particularly important to ensure that the surrounding steps do not interfere with - or disrupt - the *Capacity* of the *Constraint*.

On a conventional *Kanban Board* this is achieved by looking for starvation and the busting of *Column WIP Limits*: those are the typical signs of *Special Cause Variation* (SCV) hitting the *Work Process* represented on the boards.

If we eliminate *Column WIP Limits* with the *DBR board*, we still need some way to detect extemporaneous fluctuations in the *Flow* of work. All of this is addressed by the *TameFlow Throughput Management Board* - or *TameFlow Board* for short.



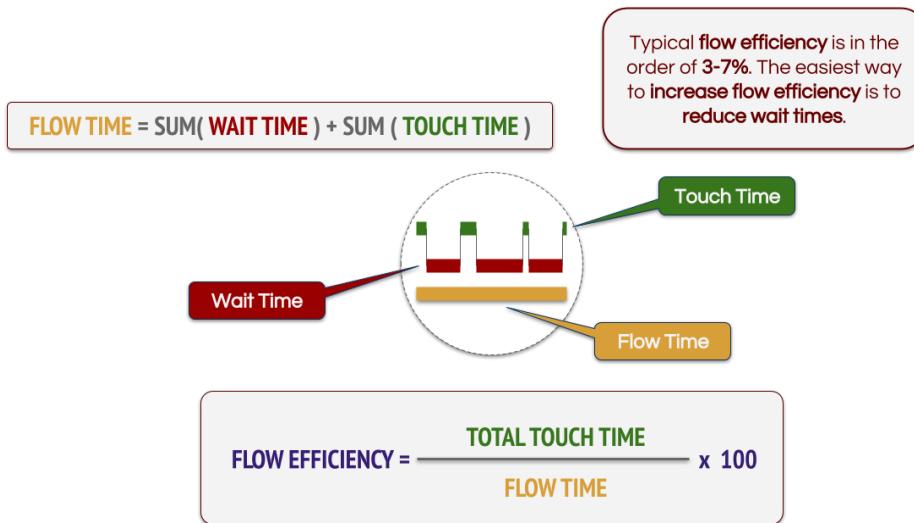
TameFlow Throughput Management Board: A *DBR Board* that is enhanced with a limited supply of *Kanban Tokens* to implement a *CONWIP* system. It is further instrumented with *Replenishment Tokens*. The relative balance of *Kanban Tokens* and *Replenishment tokens* upstream and downstream of the *Constraint* in the *Work Process* can provide leading indicators of adverse variability affecting *Flow* on those parts of the *Work Process* before and after the *Capacity* of the *Constraint* is affected.

Let's now examine these boards in more details.

Flow Efficiency Board

As we learned in *Chapter 3 - The Business Value of “Flow Efficiency”*, the *Flow Efficiency* metric is one of the most important to consider in order to determine whether an improvement is effectively contributing to the bottom line of our business.

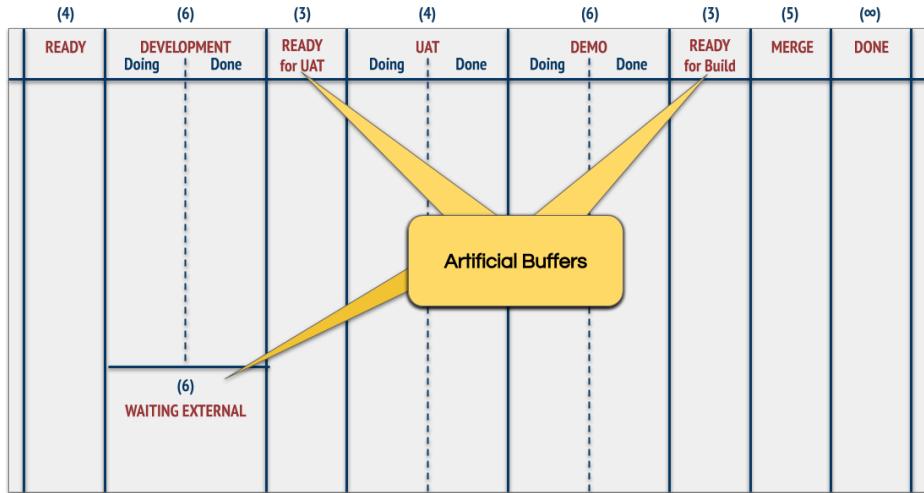
As a reminder - and in order to handle *Flow Efficiency* - we must be able to break down *Flow Time* into its constituent components of *Touch Time* and *Wait Time*.



Once we have the two components, *Flow Efficiency* is simply the ratio between all *Touch Time* divided by the end to end process *Flow Time*, expressed as a percentage.

Conventional *Kanban Boards* - designed according to the precepts of the *Kanban Method* - do not allow capturing these constituent data components in an effective way. In fact, many such *Kanban Boards* introduce *artificial wait states*, often masquerading as “ready for” and “hand over” buffers, which distort the *Flow Efficiency* metric.

For example, let us assume that Team A's conventional *Kanban Board* originally has the following layout:



Besides the presence of the ominous *Column WIP Limits* (written within round parenthesis on top of each column), we notice that the board's design explicitly contains columns and areas deliberately designed to make *Work In Process* wait. Conventional *Kanban Method* boards have at least five reasons that justify such waiting buffers and parking areas:

1. “Waiting for External” buffers. Management of releases (accumulating “big batches”).
2. The “Done” buffers.
3. *Bottleneck* mitigation.
4. To protect the (presumed and perceived) bottleneck by adding a buffer in front of *Non Instant Availability* (NIA) areas.
5. To protect the bottleneck by adding a buffer in front of a CCR (*Capacity Constrained Resource*) - when a resource is unable to do more work.

Note: The definition of a *Capacity Constrained Resource* in the Theory of Constraints is different: Any resource that, if its capacity is not carefully managed, is likely to compromise the throughput of the organization. It is what would become the constraint the moment that demand becomes greater than its capacity. A resource not being able to work, as per the KMM definition, is not exactly what is meant by TOC.

These waiting areas are not only detrimental for and distort the calculation of *Flow Efficiency*, but also give the counterproductive and wrong psychological justification that it is OK for work to stand still

waiting. This is really a sign that a cultural and mindset change is urgently needed: the boards should be there to uphold the idea that work has to flow in an *uninterrupted* and *stable* way.

In high *Flow Efficiency* or high *Throughput* environments, waiting columns; “Done” buffers; parking areas; class of service; swim lanes; and *Column WIP Limits* are all considered counterproductive to true flow - and even worse - promote a complacency culture that is detrimental to achieving high performance.

Furthermore, any time buffer columns, swim lanes, parking areas, etc. are added or removed we are actually *tampering* with the system, making it unstable and *interfering* with the optimal and natural flow of work.

Note: “Tampering” typically happens when we take action with the good intention to improve the system on the detection of *Special Cause Variation*, and mistakenly believe it is *Common Cause Variation*. It is overreacting to the detection of variation, and it will paradoxically induce even more variation while increasing costs, *Work In Process*, and *Flow Time*. Tampering is one of the worst anti-patterns, and yet it is extremely common as a consequence of using *Column WIP Limits*, buffers and parking areas. The *TameFlow Approach* is designed to actively counter these tendencies.

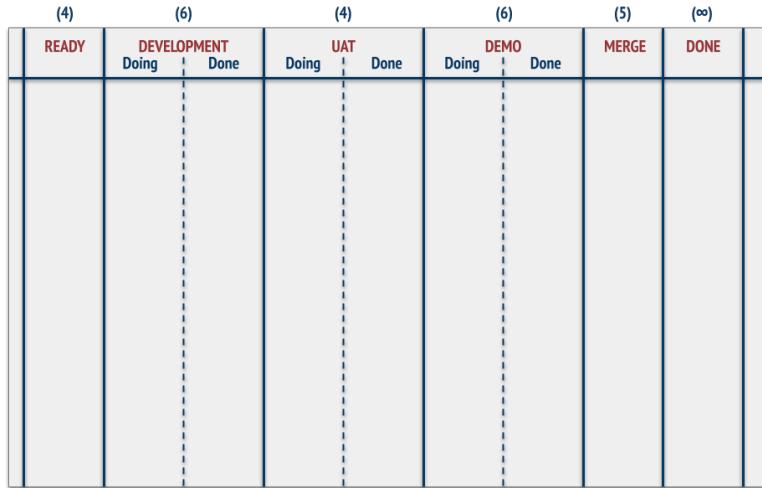
Having said that tampering is to be avoided (especially by over adjusting *Column WIP Limits*), this should not be understood as a guideline for dodging adding or removing columns to a board. After all, the whole point of these boards is the flexibility they afford in discovering and/or adapting to the actual flow as we go. They need to evolve to faithfully capture and represent the actual process that is used to perform the work.

Note: While the use of swimlanes to handle different *Class of Service* is not recommended, it can be judicious to use taxonomies to keep metrics by *Work Item* types and *Class of Service*. This can be achieved by using appropriate codes affixed to a *Work Item* without the visual fanfare and artificial barriers to natural *Flow* that is so typical of the *Kanban Method*. Such codes can be used for advanced analytics of *Flow Metrics* with respect to different types of work.

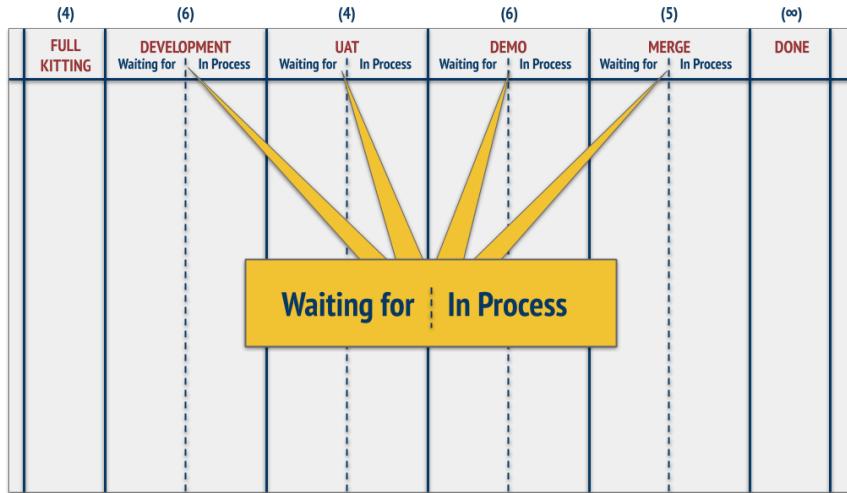
Mind you, all these elements of conventional *Kanban Boards* will give you prodigious leverage over traditional project management methods, and even over all Agile methods combined. Yet they fall short of going the distance compared to *Tameflow Kanban*.

With *TameFlow Kanban*, we will achieve fantastic results over conventional *Kanban*. The primary element that allows us to bring to fruition these outstanding benefits necessarily requires a change in culture, mindset and attitude over what is the norm in mainstream *Kanban*. One way to facilitate such changes in culture, mindset and attitude is through the adoption of operational practices and clever re-design (layout) and instrumentation (for the proper collection of flow metrics) of the existing boards.

The first required step to transform a conventional *Kanban Board* into a *Flow Efficiency Board* is to eliminate all artificial waiting columns or parking areas. The original board could be redesigned as follows:



Notice how the artificial hand-over buffers, ready for and waiting areas have been removed. It is important to highlight that such hand-over buffers are often declared as absolutely necessary. In reality, they are an artifact of *Mental Models* that do not give priority to performance but to inventory building. If the organization makes it a priority to care for performance, there is almost no process that cannot be executed without any such hand-over buffers. (It is in these instances that collective *Psychological Flow* comes into play too.) The next change is subtle - almost cosmetic - but with deep consequences both on how metrics are collected, and on how the people running the process perceive the information given by the board. The revised board, which is a proper *Flow Efficiency Board*, will look as follows:



There are two changes:

1. The *Ready* column has been renamed to *Full-Kitting*.
2. All columns on the board are divided in two semi-columns: *Waiting for* and *In Process*.

Note: Some astute readers might also make this reflection. *Flow Efficiency* boards better represent the *Value Stream* of most processes. When you arrive at a restaurant or at a hospital you are not in put in a *Doing* state upon arrival. First you identify yourself according to some procedure; then you *Wait* and only after a while are you *In Process*. Conventional *Kanban Board* design seem detached from that reality.

Full-Kitting

The first change is based on the premise that before we even start working on any item, we want to ensure that everything that is needed to get the *Work Item* to the final *Done* column is readily available. This is a different concept than that communicated with *Ready* and entails a different activity.

Full-Kitting is more powerful than *Ready*. *Ready* typically means that the *Work Item* is ready for the next immediate step (column). *Full-Kitting* means that everything that is needed for the whole journey to *Done* is known, available or can be made available just-in-time when required. *Full-Kitting* is similar to a pre-flight check: unless all checks are green, the aircraft will not take off. If things are not available,

then it is better to keep the item *out of process* and leave more options open - which is another form of *Postpone Commitment* and hence avoids unnecessary *Work in Process*.

This is a particular significant reflection. If an item is ready for the next immediate *Development* column, everything that is needed for the entire journey might not be. For example, if the team is developing software for a car, and the user acceptance test needs to be done in icy conditions, then *Full-Kitting* would ensure that those icy conditions be made available by the time the software arrives to the *UAT* step. Either start of work is postponed until the seasonal conditions allow for the testing (assuming it will be a cold winter), or the whole prototype could be shipped to the mountains to find the right testing conditions. Unless the full journey conditions are met, the *Work Item* is not released into the process. (This of course goes against the Agile precept of starting work with imperfect information, which causes *WIP* increases with all the ailments associated with that condition. Another flawed *Mental Model* in perspective).

Friction is required in knowledge-work and the *Waiting for* and *In Process* columns subtly create a psychological conflict as positive tension becomes a performance driver. Those *In Process* need to avoid overburdening and require a clear path to *Done* and do not want to appear to make the entire system *Waiting for* them.

Yet, because everyone will be made aware of the significance of *Constraint Management*, there will also be the awareness that there is no dishonor in becoming the Herbie of the situation. On the contrary, there is the confidence of knowing that if that were to happen, then the *whole organization* will step in to help (i.e. Step 3 of the 5FS: “*Subordinate to the Constraint*” and Step 4 “*Elevate the Constraint*”).

Explicit *Waiting for* and *In Process* Columns

The second change appears cosmetic but is of paramount importance for the cultural/mindset change that needs to happen. We want to accurately reflect what matters for the sake of *Flow Efficiency*, and have both visual and verbal triggers that motivate us to care about *Flow Efficiency*. We need to have *Wait Time* and *Touch Time* explicitly visible in the process, and we actively design the board accordingly.

Note: *Flow Efficiency* deals with *Wait Time* and *Touch Time* that happen during the *Flow Time of the process*. Any “waiting time” spent outside the *Flow Time* of the process (i.e. “queue time” or “delivery time” should *not* be included in the calculation of *Flow Efficiency*, though mainstream Kanban practitioners often fall into this trap by habit, and consequently distort the *Flow Efficiency* figures.)

At first glance, it seems that the mere renaming of the partial *Done* semi-columns into *In Process*, and the *Doing* columns into *Waiting for* - especially for the practitioners of the *Kanban Method* - will appear awkward.

However, things are more structural and sane.

First, each and every column that implies active work, i.e. where there is *Touch Time*, is split in two semi-columns: an initial and passive *Waiting for* and a final and active *In Process*. Notice, for instance, that even the original single *Merge* column is now also split this way.

This layout and naming convention helps us to think about the work process as an alternation of *Wait Time* and *Touch Time* semi-columns - and this conforms more with the ideas we need to care about to improve the *Flow Efficiency* metric. It also gives a clear place where *Wait Time* metrics can be tallied up and measured more precisely for the computation of the actual *Flow Efficiency* figures.

The change is structural as well, because while in the conventional board the passive *Done* semi-column comes *after* the active *Doing* semi-column of the same activity (i.e. *Development*), in the *Flow Efficiency Board*, the passive *Waiting for* semi-column comes *before* the corresponding active *In Process* semi-column.

There is visual anticipation, expectation, tension, friction and psychological drive built in the design and vocabulary of the board. Additionally, the *Kanban Method's* “*Done*” semi-column is designed for hand-over work from one step or process to the next; so it is not designed to capture *Wait Time* metrics.

This column renaming and the restructuring of the board does not change the nature of time from the perspective of a *Work Item*. When a *Work Item* is waiting, it is... waiting! But the new terminology changes the perception (*Psychological Flow*) for the workers.

The intent is to make them gain ownership and responsibility for the *Wait Time* associated with their part of the process. And for those *Work Items* that are *In Process*, the team shouldn't pull work from the *Waiting For* semi-column prematurely into the *In Process* semi-column, because they still need to limit WIP.

It is clear that for this to be really effective, it is necessary to work with the mindset and attitudes of the people; and to make it a prime imperative that *Work in Process* must get done as soon as possible. This is all part of the *Psychological Flow* of TameFlow. People will start to act differently when *Work Items* drop into “their” *Waiting for* columns. At that point, what matters is execution; what people do; how they do it.

Note also the choice of calling the active column *In Process* rather than resorting to the more common term of “*In Progress*” - we want to highlight that something might very well be in “process” (inside the *Work Process* and not waiting for workers), but still be stuck and not making any “progress” at all.

Let us also not forget the nefarious effect of *Cost Accounting* drivers in all of this. With a *Done* column, people will get more complacent and relax. When *Work Items* drop in a *Done* column after their *Doing* column, there will be a sense of accomplishment. Conversely, the folks in the next step will see things piling up in the previous activity’s *Done* column and subconsciously think: “*Oh good... I have more stuff coming my way... I don't have to worry about appearing idle while I am waiting for the next piece of work.*” Clearly this is a mindset induced by a *Cost Accounting* culture, where it is imperative to keep people busy at all times. In the *Cost Accounting* world, wait time means “resource efficiency;” it implies an inventory that is valued when accumulated, like an asset.

Again, it is a mindset issue.

Wait Time inside the process is always indicative of work waiting for workers. But what do we (and actually everybody in the organization) do about it? What do we understand it to be? And do we consider it as an asset or a liability?

If instead people have a *Waiting for* semi-column at the start of their activity column, the reaction is different. They might subconsciously think: “*Oh no! I need to get my stuff done as soon as possible, so I can take care of the next thing waiting there for me!*”

If team members have been informed about the significance of *Flow Efficiency* they will be more careful not to let work wait while “*In Process*.” This reflects the *Throughput Accounting* perspective, where greater *Wait Time* translates to lesser *Throughput*. Higher *Wait Time* implies inventory that is piling up and that is a burden, a liability.

In fact, all those “buffer/ready” columns and other parking areas do not change the *nature of Wait Time*, but they do have a profound impact on how we may interpret and relate to *Wait Time*. They have a deep psychological impact on the way knowledge-workers go about working and making decisions about what really matters and what is of the highest priority.

In particular, we do not want unstated, tacit, or implicit decision-making criteria to be exercised so that they actually create more and unnecessary *Wait Time* (with “parking space” columns), though that is exactly what happens with the *Kanban Method*.

Managing Flowbacks

There is another aspect that becomes self-fulfilling as the system becomes more and more unstable due to induced artificial *Wait Times*. It is the appearance of defects, and the need of “rework” which compounds into more “unplanned” and “induced” work.

When a *Work Item* is moving along its *Work Process*, there are instances where it needs to be sent back “upstream” for action. The work *flows back* to an earlier stage, typically for some corrective or additive action.

Moreover, notice that one reason to introduce the initial phase of *Full-Kitting* is to avoid unnecessary *Flowbacks* that often are due to some crucial piece of information missing. *Full-Kitting* is one way to mitigate the occurrences of flow backs, but they cannot be avoided altogether.

So it becomes important to have sound policies for handling flowbacks.

Flow Efficiency Inflation with Flowbacks and Conventional Kanban Boards

We will shortly explain how *Flowbacks* should be handled from a performance focused perspective. But first, it is useful to reflect on the fact that the conventional way of structuring process activities in a pair of *Doing* and *Done* semi-columns really creates both a psychological and a measurement problem with *Flowbacks*.

To begin with, there is a logical problem: conventional *Kanban Boards* with the *Done* semi-columns do not allow us to manage *Flowbacks* in a sensible way.

For example, if the *UAT* activity detects that further work is required by the folks in charge of the *Development* activity, they will send the defective *Work Item* back upstream to the *Development* activity. But where can they put that item? Certainly it does not make sense to land it in the *Done* semi-column, because that piece of work is not “done” or there would be no need to send it back upstream to begin with. So they end up depositing the *Work Item* in the *Doing* column of the upstream activity. This seems to work and is innocent enough at first sight.

This *Flowback* custom works correctly (for that defective *Work Item*) *only* if the upstream activity people interrupt what they are doing and immediately pick up the defective item. Why? Because if the defective item lands in the *Doing* semi-column and stays there waiting for people to become available to correct it, there is a logical error. The item is *waiting for the workers*, but it is *placed* in an active *Doing* semi-column.

If automated tools are used to collect *Flow Metrics*, the *Flow Metrics* would be distorted (if *Wait Time* and *Touch Time* are recorded at all, which seldom happens). We find ourselves recording *Touch Time* while in reality the *Work Item* is experiencing *Wait Time*. In short, *Touch Time* would be artificially inflated, and the resulting *Flow Efficiency* figures would appear to be much better than what they actually are.

Notice that the same problem would happen even when the upstream workers interrupt their current work to immediately take care of the defective element. Why? Because then the inflationary *Touch Time* would be counted against the *Work Item* whose work was interrupted. The workers cannot put it in “their” *Done* semi-column because the work is not “done” at all, and they will not put it in the previous state’s *Done* column for the same reason. They will keep the item in their *Doing* semi-column, even if in reality that *Work Item* is waiting - because it has been set aside and is not being “touched.”

In either case, the reported *Flow Efficiency* figures will be inflated!

And since *Flowbacks* are so frequent (in general and especially when *Flow Efficiency* is low), this might explain why the *Kanban Method* reports achieving *Flow Efficiency* figures of 40% and more, which are truly unattainable in knowledge-work.

Flowbacks with Flow Efficiency Boards

Let's now consider how *Flowbacks* could be handled with the *Flow Efficiency Board*. Note we say "could" be handled, because in reality in the *TameFlow Approach*, *Flowbacks* are despised as we will see shortly. In any case, if you are not adopting the *TameFlow Approach* in its entirety, we can examine how *Flow Efficiency Boards* are still more advantageous for handling *Flowbacks*, in case you really need them, instead of conventional *Kanban Boards*.

If a defective item is sent back to an upstream activity for corrective intervention, that *Work Item* will land in the *Waiting for* semi-column of that activity. Should it be so important (which it would be normally) to require the immediate action of the workers of that activity, the workers would place the *Work Item* they were currently working on into the *Waiting for* column before commencing work on the defective *Work Item* which now stands in the *In Process* column.

In either case, *Wait Time* and *Touch Time* would be accounted for correctly, both for the defective *Work Item* as well as for the *Work Item* that was currently being worked on in the upstream activity.

How to Manage Flowbacks in TameFlow

As we have just seen, *Flow Efficiency Boards* are well equipped to handle *Flowbacks* (unlike the conventional *Kanban Boards*). In *TameFlow* we disallow *Flowbacks* from happening at all. Instead of sending any defective *Work Item* upstream, we call the corrective work force to come downstream. The workers of the needed upstream activity will interrupt their current *Work Item*, put it in "their" *Waiting for* column, and join the workers at the downstream state where the defect was detected. In Agile terms, they would "swarm" on the problematic *Work Item* where it happens to be.

This makes sense on four counts.

First, if the items were released into the *Workflow* according to some prioritization criteria (whereby more important *Work Items* enter the *Work Process* before less important one) it then becomes obvious that the downstream *Work Item* should have a higher priority than the upstream one. Hence, if work needs to be suspended on one of the two items, it is more appropriate to do so on the upstream *Work Item* which is less important. (Though, beware, there might be exceptions to this, as we will see in more detail in *Chapter 18 - Execution Management and Governance in PEST Environments*, where due to abnormal ageing, less meaningful *Work Items* might "overtake" more important ones on the board.)

Note: This reasoning is not applicable if, as we suggest in *Chapter 15 - Outcomes, Values and Efforts in PEST Environments*, we fully adopt the concept of a *MOVE*. In such case, all *Work Items* belonging to the *MOVE* have the same importance. However, what we describe here is entirely applicable compared to a conventional situation, where ordinary *Kanban Boards* are employed.

Second, it is more attuned to the *Psychological Flow* dimension of *TameFlow*. When work has started, the clock starts ticking for that *Work Item*. The *TameFlow* attitude is: "*Once we start working on an item, we will bend over backwards to get it done. Once the race is on, it is on. You have to get to the finish line. If you stop to tie your shoelace, the race doesn't stop and wait for you!*"

Third, this favors the practice of *One-Piece-Flow*. It is important to understand that *One-Piece-Flow* is not about a "single" or "one" piece that moves, as most authors wrongly state. It is about the *Flow* of that one *Work Item* that needs to be a single, uninterrupted flow. In knowledge-work it is impossible to weed

out *Wait Time* entirely. Yet, if we touch a *Work Item*, then we prefer to touch it once, and only once, and for as short a time as possible. If we have to stop touching it, then we pull the *Andon Cord* and get those concerned to work on the problem together and immediately!

Fourth, if the upstream people are keeping busy for themselves (drinking coffee...) and don't react to the call for help from the downstream activity, then there will be *Work Execution Signals* that will be raised to get attention. First, the *Work Item Ageing Signals* of the downstream blocked *Work Item* will eventually cause a yellow or red signal on the *Work Item* itself. Then, if the waiting persists even further, we will get a yellow or red *Work Execution Signal* on the *MOVE Buffer*.

Note: We have not yet described *Ageing Signals*, *Work Execution Signals*, *MOVE Buffers*, nor the *Bubble Fever Charts*. We will encounter them later in *Chapter 18 - Execution Management in PEST Environments*.

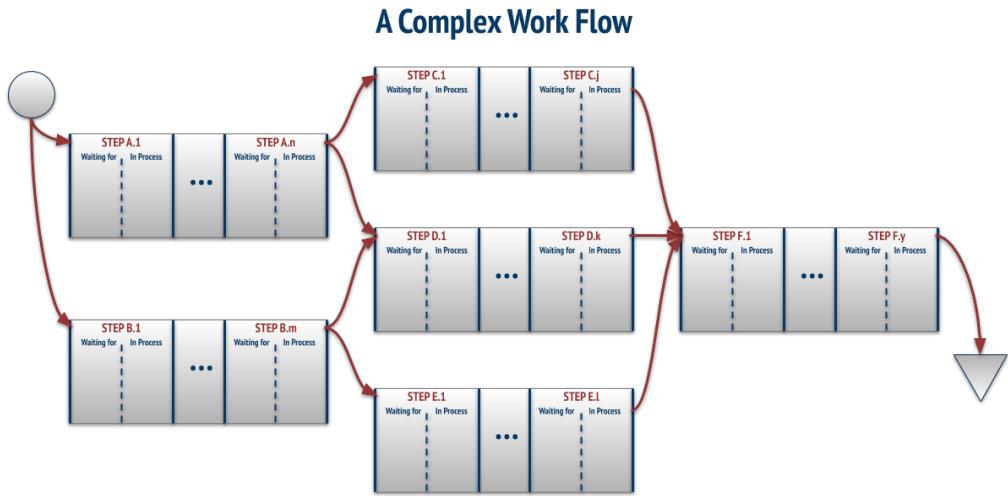
At that point the issue is escalated to the unit/department head, who will intervene by focusing on the cause of the signal. If even that is insufficient, then ultimately the signal will show up on the *Portfolio Bubble Fever Chart*, where the corresponding red bubbles will be the most critical ones. Then the entire organization will focus on the cause of the signal - with the blessing of all top managers as well - and all the coffee drinkers will get a wake up call.

This is why it is so important that the whole organization - and not only the team - understands why we have these signals, these leading indicators. The *Mental Model* has to be the same everywhere in the organization so that there is always unanimity and no compromises about what is the "right" thing to do, and where focus and action should be.

With this underlying attitude the *Kanban Boards* can become much simpler and linear, without preferential lanes, blockage areas, ready for queues, hand-over buffers and other decorations that just add information overload to no good effect, as we often observe in conventional *Kanban Boards*.

Flow Efficiency Boards and Complex Work Flows

The layout of *Flow Efficiency Boards* is also more useful in a *PEST* environment, where workflows might split, diverge, and then converge again.



The entry point of a *Work Item* moving out from the *Work Process* of one stream into the *Work Process* of another stream should, naturally, be a *Waiting for* semi-column, rather than a *Doing* semi-column, for the same reasons described earlier.

Flow Efficiency Boards give us more flexibility in complex situations, not only when the *Work Flow* may split into different streams or teams, but also when there might be multiple equivalent individuals or teams that can pull a request.

Imagine having one *Waiting for* column for a *Test* activity, where you might have three distinct test teams (with corresponding *Kanban Boards*). The single *Waiting for* column could be a single waiting queue in front of the three teams. This is very similar to what you might see at some check-in counters at the airport: one single queue and a number of counters and clerks.

Drum-Buffer-Rope Board

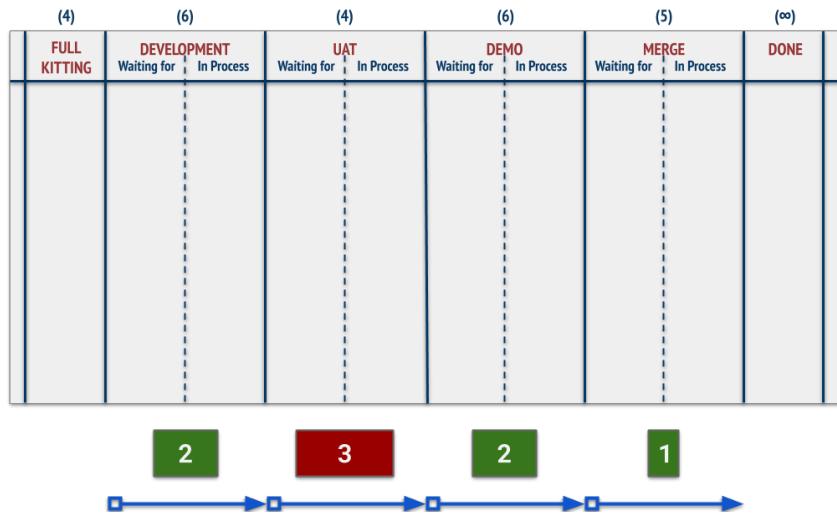
The *Flow Efficiency Boards* introduced above will allow us to correctly account for *Wait Time* and *Touch Time*, and hence compute *Flow Efficiency* correctly. It will also allow us to handle *Flowbacks* properly, if we still want to use the conventional idea of moving defective work upstream for corrective actions. (Though, as we said, with *TameFlow* we prefer never to send work upstream. Instead, the appropriate workers move downstream and swarm on the problematic *Work Item* while it remains in the *Touch Time* of the activity that detected the defect). These are all significant motivational, visual and psychological improvements over conventional *Kanban Boards*.

Yet there is another - even more important - aspect of *Flow Efficiency Boards*. Namely, they allow us to identify and manage the *Constraint* in the *Work Process*.

Using Flow Efficiency Boards to Identify the Constraint in the Work Process

The *Constraint* in the *Work Process* is identified simply by looking for the activity column that has the highest average *Flow Time*. (And it doesn't matter if that activity is performed by an individual, a pool of individuals, or collaboratively by multiple individuals working as a team.) Conventional *Kanban Boards* will reveal the temporary *Bottleneck* in the *Work Flow* by making visible queues and starvations on the board. Such queues and starvations are produced primarily by *Special Cause Variation*. And, as noted earlier, they are even induced by the inconsiderate use of and tampering with *Column WIP Limits*. The greatest performance improvements can be achieved in a *systematic* way only if we are able to identify the recurring sub-optimal conditions that are produced by *Common Cause Variation*. One way that *TameFlow* employs to reason about *Common Cause Variation* is by looking at the behaviour of the *Constraint* in the *Work Process*. Thus, there is a need to be able to identify where this *Constraint* happens to be in the *Work Process*.

Note: While queues and starvations in the *Work Flow* can be subject to lots of fluctuation and be victims of a lot of instability, the location of the *Constraint* in the *Work Process* is much more permanent. That is why improvements that address *Common Cause Variation* have such a huge impact. The *Kanban Method* deals with variation at the “qualitative level” only and is fundamentally oblivious to *Common Cause Variation*. Tangible improvements only occur by luck when the improvement actions just *happen* to affect the *Constraint* in the *Work Process*!



For example, we can explicitly add *Average Flow Time* boxes at the bottom of our columns, and highlight where the *Constraint* in the *Work Process* is located. In this example it is the UAT column that has the highest average *Flow Time* of 3 days; hence, we can consider the UAT activity as the *Constraint* in the *Work Process*.

It is important to see how things fit together. The conditions of applicability of *Little's Law*; the focussing on *One-Piece-Flow*; the limitation of *Work in Process* (via *DBR Scheduling* that we will see shortly); the reduction or elimination of *Multitasking*. All are elements that contribute to creating stable settings whereby the collection of *Flow Time* metrics becomes more dependable, since most sources of artificial and self-inflicted disturbances are eliminated. All of this contribute to establishing those stable conditions that allow us to detect the *Constraint* in the *Work Process*. It is the *Jeep* that we want to keep in top trim.

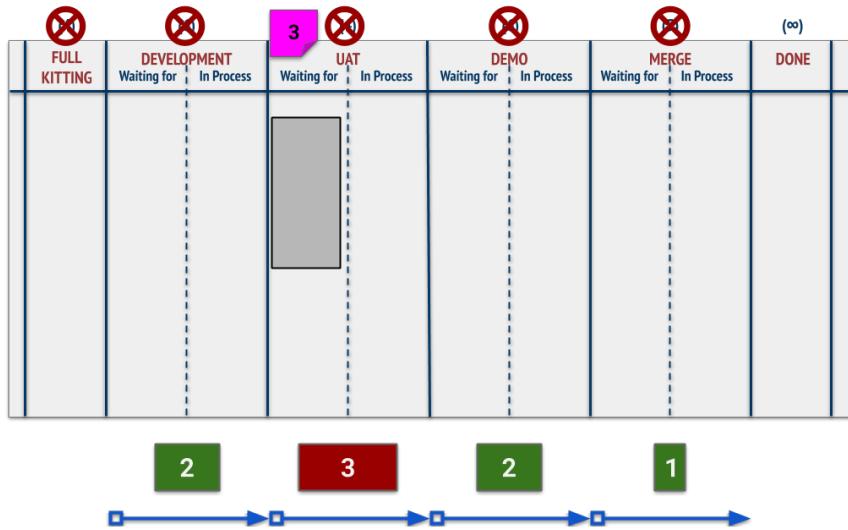
Note: A refinement of this thinking - which would be even more precise - is to consider the average *Flow Time* of the *In Process* semi-columns of all activities. In other words, we would be looking at the average *Touch Time* of the activity. Conversely, one could also focus only on the *Waiting for* semi-columns alone, because - due to the low *Flow Efficiency* conditions - they capture the bulk of the time spent in that process state and effectively represent the average queue length (i.e. average *Wait Time*) in front of that process activity. For practical purposes (especially for ease of illustration on these pages), using the average *Flow Time* of the whole activity rather than the average *Touch Time* or average *Wait Time* is often good enough. These are alternatives to keep in mind and a worthwhile model to consider to gain deeper insight into our processes.

From *Flow Efficiency Boards* to *DBR Boards*

Once we know which activity is the *Constraint* in the *Work Process*, we can transform the *Flow Efficiency Board* into a *Drum-Buffer-Rope Board*. The change is feasible in three easy steps:

1. Eliminate all *Column WIP Limits*.
2. Consider the *Waiting for* semi-column of the activity that is the *Constraint* in the *Work Process* as the *Buffer* of the *DBR Scheduling* mechanism.
3. Determine the minimum size of the *Buffer* and make it explicit.

The updated board could look as follows:



Note: In order to distinguish this kind of *Buffer* that we place in the *Waiting for* semi-column of the activity that is the *Constraint* in the *Work Process* from other kinds of *Buffers* we will encounter later, we will refer to it as the “DBR Buffer.”

Using *Column WIP Limits* is one of the most counterproductive contemporary practices. While they do help to bootstrap *Flow* because they limit *Work In Process*, they are more like training wheels. Instituting *Column WIP Limits* while at the same time striving to increase *Flow Efficiency* makes as much sense as having payrolls and speed bumps every other mile on the Autobahn: they will prevent you from moving fast!

Eliminating *Column WIP Limits* is essential to realize high flow performance. Eliminating *Column WIP Limits* does *not* mean that we don’t care about *limiting Work in Process*. The whole idea of *Flow* is to keep *Work in Process* as small as possible, to maximize the *Throughput* the system is capable of.

It is possible to limit *Work in Process* in a much more effective way: namely by using the *Drum-Buffer-Rope* (DBR) scheduling mechanism which is so natural when thinking from a *Constraints Management* perspective.

DBR Scheduling provides the optimal way to minimize *Work in Process* and maximize *Throughput*. It is the same as the one introduced in *Chapter 12- Drum-Buffer-Rope Scheduling*, though it was described in relation to the *Constraint* in the *Work Flow* rather than the *Constraint* in the *Work Process*.

We will have an explicit *Buffer* with an explicitly stated *Buffer Size*. The intent is to always have at least that number of *Work Items* in that *Waiting for* column that is acting as the *Buffer*. If there are more *Work Items*, we would simply not release any new work into the system until the number of *Work Items*

in the *Buffer* goes below its threshold.

In order to enable uninterrupted *Flow*, we must ensure that the *Constraint* is always busy. The purpose of the *Buffer* is precisely this: to make sure there is always sufficient work in front of the *Constraint*. As soon as a spot is freed in the *Buffer*, a new item is released into the *Work Process*. *Financial Throughput* is lost forever every time the *Constraint* is idle; so we must ensure it is always busy.

When a *Work Item* moves from the *Buffer* in the *Waiting for* semi-column to the *In Process* semi-column, we have a *Replenishment Signal* (the “drum beat”) that goes to the beginning of the board, and allows the next top most item of the *Full-Kitting* column to enter the *Work Process*.

The *Buffer Size* must be chosen with care: it must be sufficiently large so that as *Work Items* progress from the start of the board to the location of the *Buffer*, the *Buffer* itself is not starved. In practice, the longer the journey from the start of the board to the *Buffer*, the larger the *Buffer Size* needs to be. The *Buffer Size* will be affected by the location of the *Constraint*. The more upstream the *Constraint*, the smaller the size of its *Buffer*.

Since *Throughput* of the system cannot be greater than the *Throughput* of the *Constraint*, we can expect that the *Rate of Replenishment* should (on average) be equal to the *Throughput* of the *Constraint*. With *DBR Scheduling* we are setting the pace not on the basis of a predetermined rhythm, but on the basis of imminent capacity availability on the *Constraint*. The *Buffer Size* should simply be considered as a *time buffer* that allows *Work Items* to be released in order to reach the *Constraint* just in time as the *Constraint* itself becomes available for them.

In the example illustrated, the average *Flow Time* on the *Constraint* is three days, so the average *Throughput* is one-third of a *Work Item* completed per day. (Remember, that in virtue of *Little’s Law*, *Throughput* is the ratio between the *Work in Process* and the *Flow Time*.) We can expect to receive the *Replenishment Signal* for the next *Work Item* once every three days. (Note that depending on how badly the system is hit by *Special Cause Variation*, we may need to increase the *Buffer Size* to cater for such variability). This means that the *Constraint* can be kept busy for nine days, and that the upstream activities could experience delays of up to nine days, without the *Constraint* being starved.

The important concept to grasp here is that we are strict with releasing new *Work Items* only when we get the *Replenishment Signal*, then the only amount of extra *Work in Process* in the entire system is represented by those three *Work Items* in the *Buffer*. This is unlike the situation that is created when using *Column WIP Limits*, where a single column could be arbitrarily sized well in excess of the *Throughput* that can be sustained by the *Constraint*. This is yet another rationale why using *Column WIP Limits* would release more unnecessary *Work in Process* than would otherwise be observed with a proper management of *Protective Capacity* (see Chapter 17 - *Introduction to Full-Kitting* for more information on *Protective Capacity*).

Buffer Zones and Buffer Signals

The *Buffer* is often further divided into thirds, and the amount of *Work Items* actually present in the *Buffer* give us some actionable *Signals*, as follows:



Green: The buffer is full (*green*), the situation is deemed normal, and we have no concern.



Yellow: The buffer is only by two-thirds full (*yellow*), then we should be watching the unfolding of the situation because it could deteriorate quickly - this is a plausible *Common Cause Variation* alert.



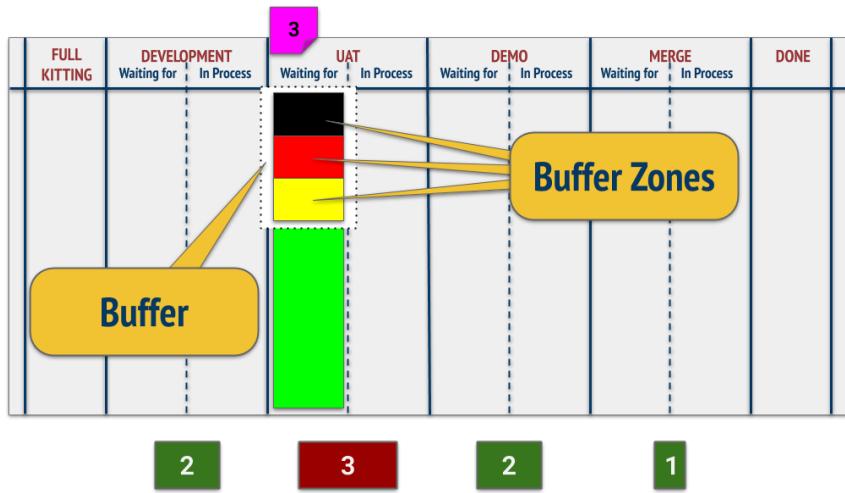
Red: The buffer is only by one-third full (*red*), then we should immediately take action, otherwise the *Constraint* could be starved - this is a plausible *Special Cause Variation* alert.



Black: The buffer is empty (*black*), then we are in an emergency situation and some drastic intervention is required - this is an “All hands on deck” alert.

Note: The use of these specific colors respects the convention of the *Theory of Constraints*. They must not be confused with ordinary RAG status reporting of conventional project management, where the interpretation of colors is different. In RAG reporting, the color represents a *status* (like a state of fact) whereas in our situation the color represents a *signal* (that is, a call to action).

After the final transformation, we can visualize a *DBR Board* as follows, where the *Buffer Zones* are represented with colors.



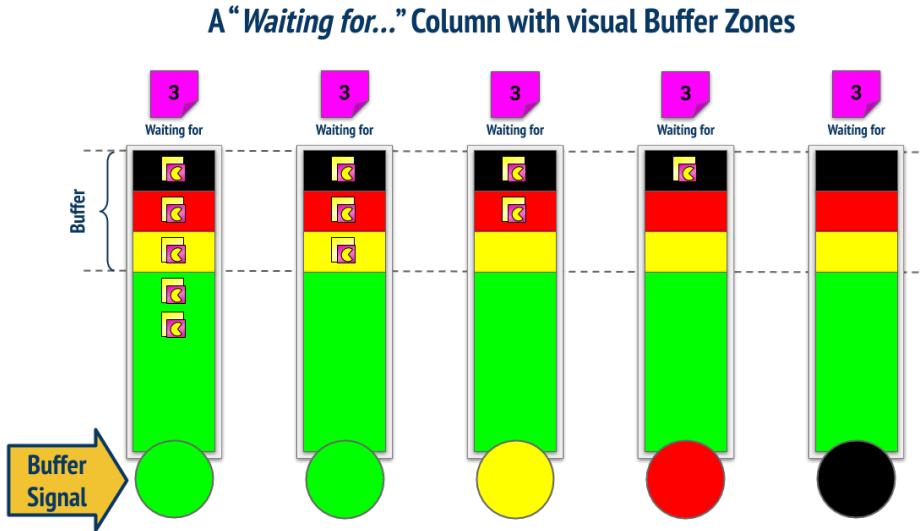
The expediency here is to physically size the *Buffer* area of the first three zones such that they can each accommodate a third of the number of *Work Items* that we want to have in the *Buffer*. In the

example, since the *Buffer Size* is 3, we will draw the *Black*, *Red* and *Yellow* zones so that they can each hold just 1 card.

Also, we color these *Buffer Zone* areas of the semi-column so that they correspond to the *Buffer Signals* that have to be read when they are uncovered and fully visible.

Of course, we will employ the operational habit to lay the cards out contiguously - with no gaps - from top to bottom - in the *Waiting for...* semi-column; and naturally in the order of decreasing priority, with the most important or urgent card on the top.

Then, as the cards move through the board we will be able to read off the signals, depending on their "occupancy" of the *Buffer Zones* as illustrated here under:



This is how we can read off the signal by looking at the top most *Buffer Zone* area that is fully visible, like this:

- When all *Buffer Zones* are covered by cards (i.e. as long as there are 3 or more *Work Items* in the example semi-column), and the *Buffer* is full: the signal is *green*.
- When the *Buffer* is only 2 / 3 full (i.e. there are 2 *Work Items* in the example semi-column): the signal is *yellow* - notice that the *yellow* area is fully visible and contains no cards.
- When the *Buffer* is only 1 / 3 full (i.e. there is only 1 *Work Item* in the example semi-column): the signal is *red* - notice that the *red* area is fully visible and contains no cards.
- When the *Buffer* is empty (i.e. there are no *Work Items* in the example semi-column): the signal is *black* - notice that the *black* zone is fully visible and contains no cards.

We can see how these simple signals already give us the means to exercise *Execution Management* in real time.

The purpose of the *Buffer* is to protect the *Constraint* from unfavorable variation. The idea is that it can be used operationally to detect unfavorable variability *early enough* to have a fair chance of intervening and mitigating the problems, before they affect the *Constraint*.

The *Buffer* does not reduce or remove variability, as some authors believe. Its primary function is to make variability detectable and manageable early enough before it produces any damage. This kind of *Buffer* is more than a padded cushion. It is an *operational monitoring device* that gives us significant leading signals of oncoming risk materialization.

Also, unlike all other kinds of buffers, it does not cost money (it actually supports generating money) and is not temporary - though it can change location if the *Constraint* moves. *Buffers* typically trade money (cost) for reduced variability. This buffer exists more for the protection of profit, than protecting against variability just for the sake of hitting project deadlines. It protects profit also by reducing variability leading to the *Constraint* (it ensures the *Constraint* is not starved of *Work*).

The *Buffer* provides a positive economic impact, considering that we add up all three components of *Throughput Accounting*:

1. TH - *Throughput* benefits because we protect *Operational Throughput (LL)*. Without the buffer, we are increasing the risk of losing *Financial Throughput (TH)*.
2. I - *Investment* will be limited to the bare minimum. Because the *Buffer* is an instrument that effectively limits the amount of *Work in Process*, it puts a natural brake on upstream processes that create and feed requirements downstream. In this sense, the *Buffer* has a long lasting effect in terms of *reducing Investment to the minimum to sustain profitable Financial Throughput*.
3. OE - *Operating Expenses* will also be kept minimal. Operations can be “right-sized” according to the capacity of the *Constraint*, with just enough *Work in Process* to provide *Protective Capacity* as to keep the *DBR Buffer* always filled. Thus, we avoid incurring excessive *Operating Expenses*.

Keep in mind that the *Constraint* in the *Work Process* can move either way because we make changes (and elevate the capacity of the *Constraint*) or because of changes in the *shape and form* (i.e. statistical distribution) of the incoming *Work Load*. In either case, we need to update the board and place the *Buffer* on the *Waiting for* column in front of the actual *Constraint* in the *Work Process*. Remember that since we are looking at the *Constraint* in the *Work Process* - which is determined primarily by *Common Cause Variation* - its location will rarely change without us being aware of what is causing the move. There can be lots of fluctuations in the *Work Flow* due to *Special Cause Variation*, but those variations will be caught as described in *Chapter 11 - Finding the Constraint in PEST Environments* and also in *Part 6* of this book.

Note: All teams must be instrumented so that it is possible to manage the *Constraint* inside their respective *Work Processes*. However, only one team - the *Constraint* team - will need to activate the *DBR Scheduling* or *Throughput Management* features of their board, and only as far as that team is actually the *Constraint* in the *Work Flow*. All other teams just need to have the readiness to do this, in the eventuality that - due to either effective elevation of the previous *Constraint* or due to significant change in the shape and form of the distribution of the incoming *Work Load* - they become the new *Constraint*. In that case, *Constraints Management* at the *Work Process* level switches from the previous team to the new *Constraint* team. The teams that are not the *Constraint* in the *Work Flow* can resort to using *Flow Efficiency Boards* rather than *DBR* or *Throughput Boards* - yet they should always be ready to switch if and when they happen to become the *Constraint* in the *Work Flow*. Finally, there is also the case where a different team becomes the *Constraint* due to the dynamic and unfavorable unfolding of its *Work Execution*. We will learn more about the *Constraint* in the *Work Execution* in *Chapter 19 - Execution Management in PEST Environments* and in *Chapter 20 - Operational Governance in PEST Environments*. All

teams should be in a readiness state to activate the *DBR Scheduling* or *Throughput Management* features on their board whenever they become the actual Herbie of the situation.

The TameFlow Throughput Management Board

In the previous section we argued about how the *Buffer* in front of the *Constraint* serves the double purpose of protecting the *Constraint* from *variation* (whether special or common) and providing operational leading signals of oncoming risk materialization.

It seems that we might be losing one of the main strengths of conventional *Kanban Boards* that are typically adept at highlighting the presence of *Special Cause Variation* (SCV). But it is far from being the case. *Column WIP Limits* - aimed at capturing SCV signals - are in fact the catalysts of excessive SCV noise defeating their own purpose.

In order to have the ability to deal with unfavorable variation without resorting to *Column WIP Limits*, we introduce the *TameFlow Throughput Management Board*.

An illustration of such a board, with its basic rules and tokens, can be appreciated in *Chapter 7 - Accounting F(r)iction*.

This kind of board is described extensively in *Chapter 19 - TameFlow-Kanban: The Throughput Focused Kanban* of the *Hyper-productive Knowledge Work Performance* book. Refer to that chapter for further details.

Later, in *Chapter 16 - Introduction to Execution Management Signals*, in *Chapter 19 - Execution Management in PEST Environments* and in *Chapter 20 - Operational Governance in PEST Environments* we will learn about even more sophisticated and quantitative ways to detect SCV and yet remain calmly in control.

Note: The *Throughput Management* board is instrumented to detect unfavorable variation that affects a single *Work Process* (i.e. team). If the impact of such unfavorable variation becomes severe, it will affect the working of the other teams as well. This kind of unfavorable unfolding of events is detected through *Execution Management Signals*, which are described in *Chapter 19 - Execution Management in PEST Environments* and in *Chapter 20 - Operational Governance in PEST Environments*.

Revisiting the Portfolio Board

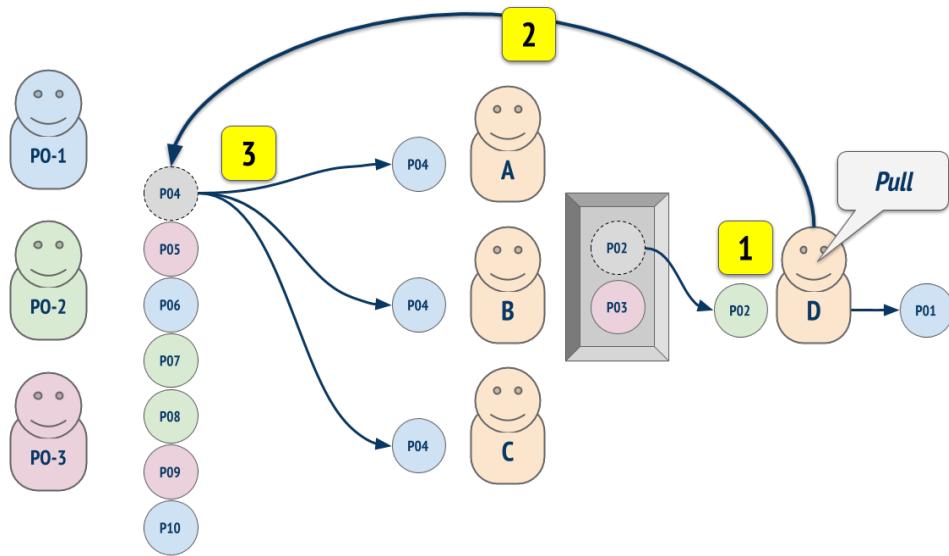
In the situation examined in *Chapter 12 - Drum-Buffer-Rope Scheduling*, Team A - the *Constraint* team - is a front-line team. It will pull its work directly from the top-most part of the *Backlog* that has been set aside to act as the *Buffer* in the *Portfolio Board*.

If the *Constraint* team were not a front-line team - but, say, Team D - the *Buffer* would become the *Waiting for* column in front of Team D. More specifically, the *Buffer* would become the *Waiting for* column in front of the *Constraint* in the *Work Process* of Team D.

The *Committed* column in front on the *Portfolio Board* would then receive the pull signals originating from Team D.

In this case, Team D would pull a new *Work Item* from its *Buffer* (i.e. its *Waiting For* column). It is that action, the movement (the “drum beat”) of a *Work Item* out of the *Buffer*, that would send a *Pull Signal* (the “rope”) to the source of replenishment, i.e. the *Committed* column in front of the *Work Flow* to feed the front-line teams. The front-line teams have capacity available because they are not the *Constraint*.

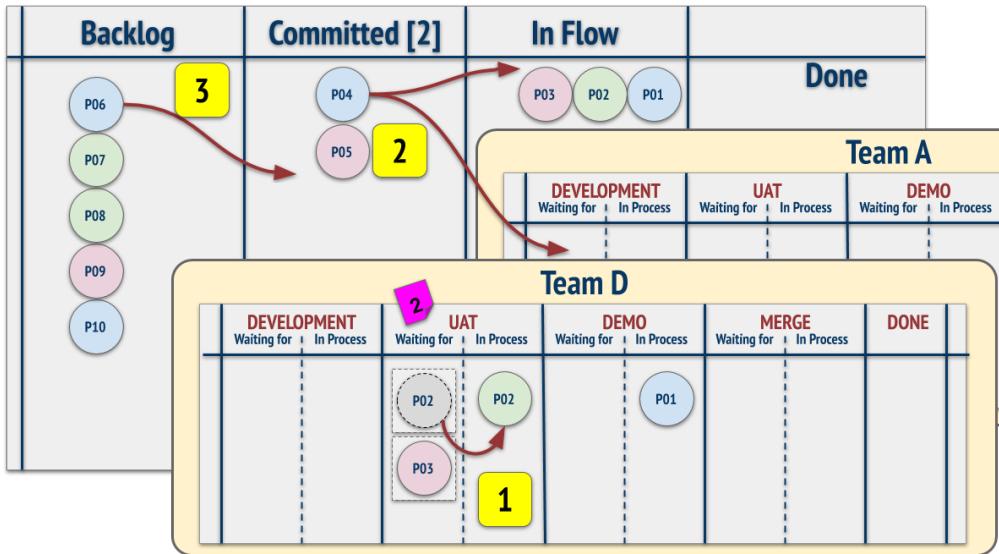
At the *Work Flow* level, we can visualize it as follows. The events labeled 1, 2 and 3 have the same meaning as we saw earlier, in *Chapter 12- Drum-Buffer-Rope Scheduling*.



In terms of *TameFlow Boards*, it would more specifically resemble the illustration below where we see Team D's *DBR Board*.

Team D's board identifies the *User Acceptance Test* (UAT) column as the *Constraint* in its *Work Process*. The UAT column has a *Waiting for* semi-column that is acting as the *Buffer* in the *Work Process*.

Notice that above it there is the *Buffer Size* indicated by the “2” on a purple sticky note. This happens to correspond to the *Buffer Size* that we have on the *Committed* column in the *Portfolio Board* (and while convenient, be aware though that the two sizes might not always be the same).



This illustration shows the following:

1. Project P02 has just been pulled into the *In Process* semi-column of the *UAT* state for Team D. This frees up a place in the buffer, and that event triggers the drum beat - the *Replenishment Signal*.
2. When the signal reaches the *Portfolio Board*, Project P04 is released into the *Work Flow*, and moves from the *Committed* column to the *In Flow* column. At the same time, the relevant parts of Project P04 are released into the *Waiting for Development* semi-column of Team A (and similarly with the other front-line teams, not shown in the illustration). Presumably, these frontline teams are *idle* (having more capacity than the *Constraint*), and thus can pick up the work immediately and move the *Work Item* to the first *In Process* semi-column of their board.
3. At the same time, the *Replenishment* from the *Backlog* is triggered. Project P06 is moved to fill in the empty spot in the *Committed* column.

Because Team D's *UAT* step is the *Constraint* in the *Work Process* of the *Constraint* in the *Work Flow*, before the *UAT* step is done, Project P04 will have had time to go through all the front line teams, including the development phase of Team D. Therefore Project P04 and will arrive at Team D's *Waiting for the UAT* column while Project P02 is still being worked on in Team D's *UAT In Process* column.

If something slows down Project P04's journey across the preceding boards, the *In Process* stage of Team D's *UAT* step would still be able to pull work and pick up Project P03, while Project P04 would catch up.

At the same time, Team D's pulling of Project P03 would release another *Replenishment* signal (because now the *Buffer* would be empty): Project P05 would be released into the *Work Flow*, and Project P07 would go from the *Backlog* to the *Committed* column on the *Portfolio Board*. Ideally, the *Constraint* would never run out of work. Should that happen, then a major escalation would be called.

As a final reflection, where you decide to visually place the *buffer* on the *Boards* is really a question of layout preference for the teams' convenience. What really matters is that the *Replenishment Signal* is

triggered by the pull performed by the *Constraint* in the *Work Process* of the *Constraint* in the *Work Flow* (like Team A originally, or Team D in this example), and that this *Replenishment Signal* causes the topmost committed Work Item at that time to be released into the *Work Flow*, then moving the most relevant backlog item to the *Committed* column (through the process of *Full-Kitting* that we will discover in Chapter 17 - *Introduction to Full-Kitting*.)

Acclamation of Idleness

In the situation illustrated, when Project P02 is pulled from the *Waiting for UAT* semi-column to the *UAT In Process* semi-column, Team D's *Development* people *have nothing to do!*

In a conventional *Kanban* implementation, this empty column would be interpreted as if the whole of Team D were starving. It would trigger a witch hunt for some *Bottleneck* upstream of Team D, to make it a target of improvement. And as usual, that initiative would be “successful” at finding such a target, and making “improvements” that - as we know - do not really matter, because they target a *Non-Constraint*.

With the *TameFlow Approach*, we know that high-performing teams will have *workers waiting for work*. This is a necessary condition to limit *Work in Process* in the presence of a *Constraint*. It just cannot be avoided. It is part of the *Subordination* step of the *5FS*.

In fact, in the situation shown, Team D is not starving at all; the knowledge-workers of the *Development* column are manifesting their *Excess Capacity* while they are waiting for Project P04 to reach them and then hand it over to the *Waiting for UAT* semi-column, just in time to keep the *Buffer* in front of the *Constraint* full. (For more on *Excess Capacity*, see Chapter 17 - *Introduction to full-Kitting*.)

Note: We might be concerned about accepting people on Team D to be idle, while Team D is the very constraint in the *Work Flow*; but then we must observe that the *Constraint* in the *Work Process* of Team D is the *UAT* column. All other columns, thus even the *Development* column, will have more capacity by definition; and hence they can afford to be idle as long as we can ensure that the *UAT* step is always busy. Of course, one way to employ the idle time, would be to help the *Constraint*, if that is possible; as we will observe in a moment.

These people are idle *by design*. Idleness - or “slack” as it is often called - is a necessary condition for high organizational performance. It has to be accepted. What should these people do? They could literally go fishing or take a swim and enjoy life during this idle time. It would not make the slightest difference on the *Throughput* and it would not increase *Operational Expenses*. Though they really need to be ready to step right back in, as soon as their pertinent components of Project P04 arrive.

They could also use their *Excess Capacity* to maybe support *Full-Kitting*, help the *Constraint*; or learn new things and skills (maybe so that they can help the *Constraint* if they are prevented from doing so because of lack of skills); or think about other improvements.

The Unresolvable Conflict between Idleness and Cost Accounting

This *Excess Capacity* is a gift that comes at zero cost. Yet, most companies would be engaging diligently in *Cost Saving* exercises and fire or reallocate people from the “oversized” development resources - basically

guaranteeing they would then become the next *Constraint* - and ensuring they can keep on firing more people in other departments now considered as “oversized.”

So, *Cost Saving* exercise after exercise, the company will deprive itself of any capacity to improve at all. But the CFOs will earn their bonuses for their stellar *Cost Saving* performances.

The *Cost Accounting* mindset dictates that *Excess Capacity* be eliminated. Hence the catch-up ability disappears. Everywhere. Consequently, the organization falls behind on any delivery expectations. Perversely, *Work in Process* starts to increase and *Throughput* decreases. Blame is attributed to uncontrollable events. The magic cure for lost capacity often becomes the pervasive use of overtime. Traditional *Cost Accounting* managers must have balanced capacity throughout the value chain, to save money, by being fully utilised and levelled everywhere. Once this is achieved, they must invest in *New Capacity* all over again, going around in circles and never becoming able to actually increase the organization's performance and bottom line results.

Cultural Impact of the *TameFlow Boards*

There is a subtle cultural change underlying all of this instrumentation of the *Kanban Boards* - and the adoption of the *TameFlow Approach* in general. In order to instill the seed of a performing organisation, we must let go of the mental model that Herbie is the cause of all our ailments. By distributing the load and adjusting to the pace, Herbie becomes the solution! Such is the nature of a *TameFlow* driven enterprise.

The signals emerging from the *TameFlow Boards* give us the immense power to zoom in on the Herbies of the situation in order to help them.

When the concepts of *Constraints Management* are first introduced - and in particular those related to *Excess Capacity* that foster idleness and slack - it is to be expected that management will resist the proposition. Naturally the resistance is based on the flawed *Mental Model of Cost Accounting*.

Embracing the methods proposed in this chapter will help retain talent, increase happiness, lead to agility and raise the bar toward management excellence.

Psychology of Wait States

An interesting effect that you will observe when adopting the *TameFlow* boards in a multi-team setting is the psychological and motivational impact of having a *Waiting for* queue in front of a *Work Process* (team). The team will be alerted that there is a lot of work to do, and management will have visual signals of where they need to focus their attention (on Herbie!).

The *Waiting for* columns allow for accurate *Wait Time* reporting and hence correct calculation of *Flow Efficiency*. Yet, at the same time, it is not in the interest of the *In Process* workers to open the flood gates and work on too many items, to get *WIP* out of control and get sub par *Throughput* because the *Constraint* becomes overloaded.

Conventional *Kanban Boards* lack such psychological and motivational visual cues that expose the necessary friction between *Wait Time* and *Touch Time* state changes.

These cues are part of the *Informational Flow* and *Psychological Flow* we come to know through the *TameFlow Approach*.

Takeaways

The *TameFlow* boards allow us to:

1. Collect the right metrics so that we can think in terms of *Flow Efficiency* via the *TameFlow Flow Efficiency Board*.
2. Identify the *Constraint* in the *Work Process* as the column on the board with the highest average *Flow Time*.
3. Support the *DBR Scheduling* technique via the *TameFlow DBR Board*.
4. Eliminate *Column WIP Limits* since *WIP* is optimized by means of *DBR Scheduling*.
5. Focus remedial action upstream or downstream of the *Constraint*, with leading signals via the *TameFlow Throughput Management Board*.

Specifically, the *DBR Scheduling* technique is of supreme importance because it extends to the *Portfolio* level and is what ultimately allows us to optimize the amount of *Work in Process* across the entire *Work Flow*, even when it involves a multitude of *Work Teams* and *Work Streams*.

The *DBR Scheduling* technique is what enables us to keep the amount of *Work in Process* as small as possible without resorting to *Column WIP Limits*, albeit without causing the *Constraint* to remain starved. It is what makes it possible to *Postpone Commitment* and *Limit WIP* and, ultimately, increase *Throughput*.

Curtis Hibbs says...

If you have ever wondered why Agile transformations result in lackluster outcomes, this is the book for you. In fact, I would go so far as to say that this book should be REQUIRED READING FOR ALL AGILE COACHES!

Even though I already knew Goldratt's Theory of Constraints, I don't practice its tenets every day and don't have its principles internalized as I should.

That is about to change because this book has breathed new life into what I should have already been doing. The Theory of Constraints is all about making changes that are effective improvements, and avoiding changes seem to make sense but have no impact (or even make things worse).

This book provides one of the clearest presentations of the Theory of Constraints and the crucial role it plays in the success of scaled Agile implementations.

Nothing gets me more excited than realizing that half of what I've been doing has been counterproductive because now I can fix it.

Knowledge is power!

Curtis Hibbs, Agile Transformation Coach and former Chief Agile Evangelist at Boeing, author of “*The Art of Lean Software Development: A Practical and Incremental Approach*.”

15—Outcomes, Values and Efforts in PEST Environments

The Virtue of Minimalism: the *Minimal Outcome-Value Effort* (MOVE)

In the ever-evolving world of software development processes and methods, one recurring theme is that of *minimalism*. Various approaches have defined concepts such as:

- Minimal Viable Product (MVP)
- Minimal Marketable Feature (MMF)
- Minimal Marketable Release (MMR)
- Minimal Marketable Product (MMP)
- Minimal Business Increment (MBI)

For some further descriptions, see the blog posts:

- “[Defining MVP, MMF, MMP and MMR](#)” by Scott Ambler and
- “[Minimum Business Increments](#)” by Al Shalloway.

These concepts are all different, yet also bear some important similarities. They all convey some notion of *minimalism*, which typically relates to the *cost*, *scope* or *time* of some business initiative. But they also imply a notion of *sufficiency* in relation to some beneficial business objective.

Note: Is it **minimum** or **minimal**? All the above terms can also be found with the adjective “*minimum*” instead of “*minimal*.” In the end, the choice of word does not matter if we understand the purpose and intent of the underlying concepts and notions; therefore, it boils down to a matter of preference. While in the past we have used the term “*minimum*,” we now actually prefer “*minimal*.” The reason is that while “*minimum*” refers to the smallest possible amount or quantity, “*minimal*” conveys the meaning of sufficiency, something that is absolutely adequate, suitable, acceptable, and fit for purpose, yet is as small as possible, and no smaller and no bigger. It is this latter meaning that we prefer to capture. Hence “*minimal*”.

What is a MOVE?

While in the previous *Hyper-Productive Knowledge Work Performance* book we used the notion of *Minimum (!) Marketable Releases* or MMR as described therein in *Chapter 20 - Improving While in the Flow*, we now extend the concept and introduce the notion of the **Minimal Outcome-Value Effort** - or “MOVE” for short.

Clearly, the aim of this terminology is to enrich the lexicon with a term that motivates direct action, and that rolls off the tongue easily. (*What is the outcome of our next MOVE? What is the value of the next MOVE? How much effort do we need for our next MOVE?*)

The idea with the *MOVE* is to be able to focus on the *minimal effort* that the organization can undertake in order to achieve a desirable *outcome* and/or produce a given *value*.

A *MOVE* encompasses all of the above minimalistic items. It can be a MVP; a MMP; a MBI; a MMF; or a MMR - likewise it can be used for their respective purposes.

A *MOVE* is also more than that. It is more generic, because it can also be used to represent:

- Objectives/Results in *OKR* formulations.
- Nodes in the Strategy & Tactics trees used in the *Theory of Constraints*.
- *TameFlow Transformational Patterns*.

A *MOVE* provides flexible, coherent, and manageable modeling and representation of organizational business initiatives and actions. *MOVEs* align with - and are consistent with - different management perspectives (top-down; value-stream; financial reporting; operations management, etc.).

In particular, a *MOVE* encompasses four co-existing perspectives, since it can be considered as:

- A unit of *stakeholder value*
- A unit of *financial throughput*
- A unit of *costing and reporting*
- A unit of *work and delivery*

Let us see who might be concerned with each one of these perspectives.



A Unit of Business Outcome

In most of the aforementioned “*de minimis*” (MVP; MMF; MMR; MMP, MBI), it is assumed that there are just a handful of stakeholders - who typically are the business owners (and sometimes the buyer of the product or service).

A *MOVE* is more generic, because it recognizes that there might be a number of different stakeholders who derive value or have an interest in the product or service under consideration. It is by satisfying the needs of these stakeholders that we recognize some business outcome.

The stakeholders may include:

- Business Owner: who typically wants to have successful and sustainable business.
- Customers/Clients: who typically pay for purchasing the product/service.
- Users: who are the actual “end users” and consume the product/service offering.
- Shareowners: who are concerned about the business soundness of the product/service offering and care about seeing a return on their investments.
- Employees: who have to “live” with the product/service being developed.
- Engineers: who need to fight the accumulation of *Technical Debt*.
- Public: who might have concerns about sustainability, fair trade, inclusion, etc.
- Regulators: who might need to see regulatory compliance fulfilled.

Note that a business *Outcome* can also be a discovery. Hence a *MOVE* will also support those business processes where there is no predefined target result, but where new information is gathered or generated during the process itself. Information that then will be used to take further business decisions. In this sense, a *MOVE* can support the market fitness discovery process such as *Customer Kanban*, *Lean Canvas*, *Continuous Innovation*, *Popcorn Flow* or *Discovery Driven Planning*. It can also be the unit of work for a *Spike* - an exploratory engineering effort - as it is known in *eXtreme Programming*.

A Unit of Throughput Value

Everything that a business undertakes must bear financial benefits. Typically, in a business setting, we would be interested in “how much money” the organization is bringing in. Those who might be interested in this aspect are:

- C-Level Executives
- Finance
- Investors

One important aspect is that we focus on the *Throughput of Value*. It is not specified that the value necessarily be financial. There are organizations that have *Goals* other than making money. For example, a hospital might be interested primarily in how many patients are treated every month. Or a charity might care about how many individuals receive help. Different organizations have different goals; yet all those goals can be quantified in *Goal Units*, and hence we can think in terms of *Throughput of Value* quantified in such *Goal Units*. With this perspective, those who have an interest in some non-profit organization could include:

- Associates
- Patrons
- Donors

In all of these cases the practices of *Throughput Accounting* can be fully applied to indicate what kind of results the initiatives might produce.

Note: From a *Goal*-oriented perspective, a *MOVE* can be considered as a *Minimal Goal Increment* - the minimal thing that can be done and that moves the organization one step closer toward the organization's *Goal*.

We also need to stress that the value produced is always considered in terms of *Throughput* as it will have to pass through the *Work Process* of the *Constraint* in the *Work Flow* or the *Constraint* in the *Work Execution*.

A Unit of Costing and Reporting

While we will primarily use *Throughput Accounting* in support of all management decisions, we cannot escape the fact that financial reporting - especially for taxation purposes - will always be based on *Financial Accounting*.

All conventional accounting and reporting practices must still be in place, unchanged. However, they can be updated to better reflect what items are actually creating value for the business. Hence the *MOVE* becomes a unit of accounting and reporting. Those interested in this perspective might include:

- Accountants
- Operations Managers
- PMOs

A Unit of Work and Delivery

Finally, a *MOVE* can be considered as a unit of work and delivery. The exact granularity of this unit can vary. In this sense, the *MOVE* is like a work package that can contain whatever elements the engineering teams decide to use to represent discrete units of work, like: *Use Cases*, *Features*, *User Stories*, *Tasks*, and so on. What matters most is that we want the content of a *MOVE* to be a *target-scope of work* (more about this in a moment).

Those interested in this perspective are:

- Product Owners
- Project Managers
- Scrum Masters
- Teams
- Suppliers

It is this particular perspective that bears the greatest weight when we need to manage outcomes; values; and efforts; and to create flow in a PEST environment. Let us see this aspect in more details.

As the name suggests, we are concerned about establishing the *minimal effort* that will achieve a desirable *outcome* and/or produce some desired *value*. The amount of work that encapsulate this effort becomes a *MOVE*.

The *Portfolio Kanban Board* will manage such *MOVEs*.

But what exactly is a *MOVE*?

A Small Target-Scope Work Package

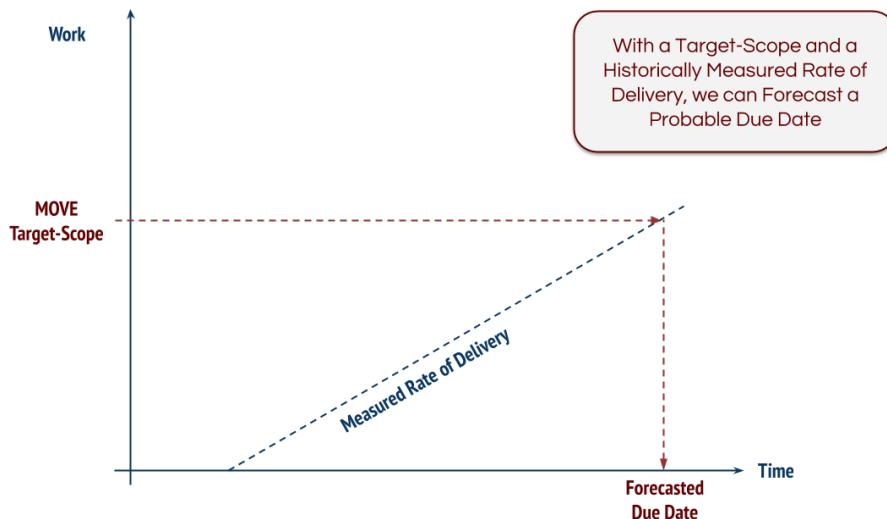
A *MOVE* is a targeted amount of work. *Target-Scope* means that the amount of work is *almost* fixed. In particular, it must be *minimal*, with respect to the *outcome* and/or the *value* that it is intended to deliver.

The minimality requirement is essential and it means that it is relatively small (not bloated), and especially supporting the lean tenet of small *Batch Size* (with all the related positive implications).

A *Target-Scope* means that there is some very well defined amount (and content) of work to aim for. It is a predetermined number of *Use Cases*, *Features*, *User Stories*, *Tasks*, or whatever units of discrete work we want to use.

Why is a *Target-Scope* important?

Because it allows us to reason about delivery in terms of a *Work/Time* diagram, like those we encountered in the first few chapters. A *Target-Scope* is simply a horizontal line (almost like a goal line) that marks off how much work needs to be delivered.



If we have a *Target-Scope*, and we have historical metrics about the system's *Throughput*, then we can project a forecast *Due Date* in virtue of *Flow Time* duration calculated according to *Little's Law*.

The above diagram is an over-simplification, because in order to effectively apply *Little's Law*, there are several statistical prerequisites, which we overlook for the sake of simplicity. Remember we are in the realm of *Mental Models*.

This gives us an idea of how it would work based on probabilistic forecasting techniques.

The Balance of Two Opposing Forces

There will be two *opposing* forces at work when striving to establish what this *Target-Scope* should be. The two forces can be described as follows:



External Forces: On the one hand, there is the desire to maximize *Outcomes and Values*. *The MOVE should be as large as possible to ensure best business fit.*

Anything more would definitely be a waste.

In short, this external force wants “*more stuff to get done!*”

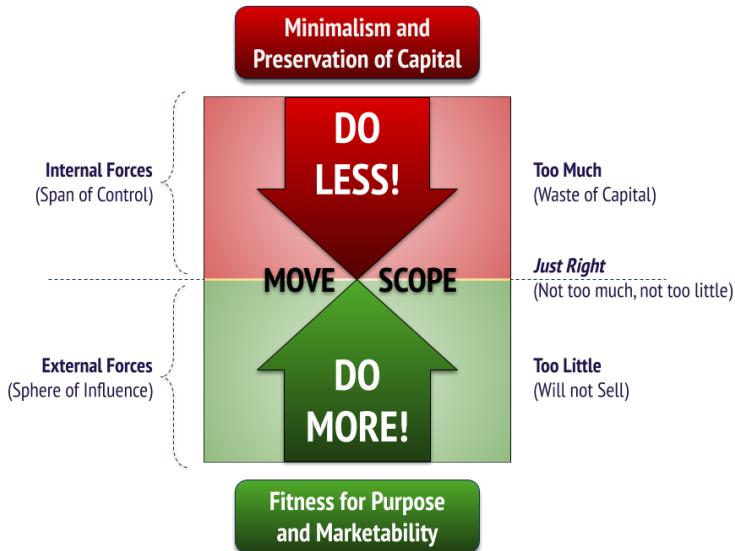


Internal Forces: On the other hand, there is a need to minimize costs and resource utilization, because businesses have limited funds. Capital must be shared and deadlines are looming. *The MOVE should be as small as possible to employ the company's capital (financial, human, material) in the best possible way.*

Anything less would definitely prevent the MOVE from delivering the desired *Outcome or Value*.

In short, this internal force wants “*less stuff to get done!*”

Thus, *Target-Scope* is the minimum scope that will allow us to attain the *Outcome and/or Value* without wasting *Effort* (in the form of financial, human or material resources).



The *Target-Scope* is defined in *economic* terms, because it explicitly accounts for the preservation of the business's capital.

This means that in any scope definition activity, *the financial side of the business should always be actively involved*. In most conventional situations, finance is involved only at the annual budget approval level, and not in the day to day operations.

This is a major difference brought by the *TameFlow Approach*.

Unit of Commitment

The *Target-Scope* is also a unit of commitment. In the portfolio *Kanban Board* we saw at the end of *Chapter 12 - Drum-Buffer-Rope Scheduling*, we observed that the buffer of the constraint in the *Work Flow* was represented by a *Committed* column.

The items that enter the *Committed* column are just that: committed to be delivered! And those items will be *MOVEs*. Work has not started on them yet, but the intention to get them done is declared and sanctioned.

This is also another reason why the *Committed* column of the *Kanban Board* should contain the least amount of items possible: we do not want people to feel overwhelmed or overstretched. Naturally this goes hand in hand with the practice of limiting *Work in Process*, and with the precept of keeping open as many *Options* as possible.

Mechanism to Limit *Work in Process*

Speaking of limiting *Work in Process*, we have another synergy at play here. Since we are defining a *MOVE* in terms of a *Target-Scope* that is limited by the balance of the two forces (of preservation of capital, and of functional expansion for fitness for purpose), we have a natural mechanism for limiting work that is released into the *Work Flow*.

Risk Managed via Time and not Scope Adjustments

Because of the *Target-Scope* nature of *MOVEs*, it is necessary to become ever vigilant in terms of explicit risk management by better managing the *time* dimension. We will see how to do this in detail in *Chapter 19 - Execution Management in PEST Environments*.

What we definitely do not want to do is resort to the popular Agile practice of “*cutting the backlog*” - that is, to reduce the scope of the work. Because the scope is determined as a balance of the two opposing forces, any such descoping would practically render the *MOVE* unfit. (If the descoping is required because too much scope was added to begin with, then it means that the two forces were not in balance.)

A Note on Agile

Agile approaches are fond of using timeboxes (like *Sprints*) and then adjust the scope. One common recommendation is to have timeboxes (or *Iterations*) as small as possible. The consequence is that any significant scope of business value needs several such *Iterations* in order to be realized, because (typically) there is only so much that can fit in a small *Sprint*.

There is no problem with an *iterative* approach, but for work to flow through the organization with high performance, interrupting the flow of work every second week (or whatever the iteration duration is)

with the replanning and rescheduling ceremonies (i.e. *Sprint Planning*, *Sprint Review/Retrospective*, *Backlog Refinement*), will introduce unnecessary disruptions, which will result in overall loss of *Throughput*.

Agile proponents claim that such an approach reduces risk, by implementing the most valuable pieces of work sooner, and the less valuable pieces of work later. The rationale is that if there is a deadline to hit, the less valuable pieces can be dropped, and the project would be delivering the best value that was possible to realize by that deadline.

Note: We use the same logic of putting higher valued items first, as we have seen in *Chapter 13 - Portfolio Prioritization and Selection in PEST Environments*. Note though that the items being prioritized are atomic and *Target-Scope MOVEs*, not arbitrary partial increments which might fit in a *Sprint* timebox, and would most likely correspond to a partial subset of a *MOVE*.

Sprint based reasoning is not compatible with the *TameFlow Approach*'s view of how businesses should operate on three fronts:

1. to maximise profits;
2. to free up time (for idleness and slack);
3. to cater for more humane working environments.

When reasoning in terms of *MOVEs*, if something can be dropped as we are approaching a delivery or deadline, the very fact that it *can* be dropped means it should not have been considered to begin with. If it can be dropped, it is *excessive* and by *definition* it is not part of a *MOVE* which is *minimal*. That is unless there was an error of judgement at the outset.

This is how Agile projects become bloated and prevent optimal value delivery. With the gullible idea that things can be dropped and yet there is value to reap - in practice - a lot of waste makes its way into the backlogs.

The solution?

To have an economic view on the projects, and to involve the financial managers in the decision-making of what is inside and what is outside of the scope.

To do this adequately, the techniques of *Throughput Accounting* and *Theory of Constraints* must be employed. In most companies - including those that are excellent in operating according to Agile principles - neither TA or TOC are considered. This is why the *TameFlow Approach* will outperform traditional *Agile* approaches on all terms: operational; financial; and humANELy sustainable work loads.

Managing Scope Variation

While insisting on the notion of *Target-Scope* might seem excessive, there are many benefits for doing so (which will become even more evident in *Chapter 19 - Execution Management in PEST Environments*).

Target-Scope is not Fixed-Scope

We must not fall for the impression that *MOVEs* are *Fixed-Scope* projects - with all the dysfunctions and inflexibility that come by via *fixed scope* thinking.

Note: Most traditional Agile proponents will instinctively fall into this ambush - without thinking any deeper - because they are accustomed to manage variability through timeboxing. It is ironic that for timeboxing to work, the premise is to have a fixed scope that can fit in a two-week *Sprint* which is then typically diluted as items are discarded, delayed or postponed when they won't fit into the timebox. The missing insight is that variability can be managed differently, as we will see in upcoming chapters.

This cannot be stressed enough: *Target-Scope* is not the same as *Fixed-Scope*.

With a *Target-Scope* we strive to *keep the scope unchanged in so far as it makes sense*. We don't consider the *Target-Scope* as fixed. We don't subject to it as if it were a divine commandment. There are situations that truly require changing scope, and we must be able to handle them.

For example, if market conditions change, some planned elements might become obsolete, or new ones might be needed in order to catch up with new functions, features or capabilities presented by competitors.

While we prefer not to change the scope of a *MOVE*, it is easy to do if necessary.

In fact, referring back to the work-time diagram above where we illustrated the *Target-Scope* with a horizontal line, any addition or removal of scope will simply move the line up or down on the diagram. All other elements of risk management (which we will see in *Chapter 19 - Execution Management in PEST Environments*) will work unchanged and simply recalibrate according to the new *Target-Scope*.

Key Insight: A *MOVE* must be considered as an *atomic* unit of value (for somebody or for a specific purpose). Atomic means that we cannot take away anything from it, or the value will suffer. Consequently, we cannot do a partial delivery, or reduce the content of a *MOVE*. Likewise we cannot arbitrarily add new things to it without re-negotiating its value components, especially to ensure its value is not diluted and capital is not wasted. However if our understanding of value changes before delivery, then we will update the *Target-Scope* of the *MOVE* accordingly.

A *MOVE* View of the Simulation

With this *MOVE* perspective, we can consider the “projects” that were dealt with in the simulation game as *MOVEs*. We can further observe that every project acts as a container, like a “work package” of sorts, because it contains different coins that are distinct pieces of work (which could correspond to *Use Cases*, *Features*, *User Stories*, *Tasks*, and so on in the real world).

Every project/*MOVE* in the simulation has a *Target-Scope*, which is the number of coins it contains.

Here, a further parallel could be drawn with the Agile lexicon: a *MOVE* is like an *Epic* while the enclosed *Work Items* (the coins) are like *User Stories*.

The *User Stories* (coins) are the units of work for the operational teams. They are the actual *Work Orders* that represent the requests of the clients or of the market, and are expressed so that they can be understood and made actionable by the teams.

It is key to ensure that *all* the *User Stories* (coins) of a *MOVE* (project or bag of coins) can indeed be *understood* and *made actionable* by the teams.

Typically, during *Full-Kitting* activities (which we will see in *Chapter 17 - Introduction to Full-Kitting*), the *Flow Manager* will interact and communicate with the teams to ensure these conditions are met.

At times, the teams will respond that a certain request is incomprehensible; too big; or missing information. Then the *Flow Manager* will have to clarify, break down, or find out what is missing.

The *Full-Kitting* activity can happen both on the client side as well as on the team side. The purpose of distinguishing between client side and team side *Full-Kitting* is precisely that - to have a defined way through which this communication can happen.

In smaller teams, this is typically the function of a *Product Owner* that works closely with both the team members and the clients. In larger organizations that produce more complex products, a *Product Owner* - or more likely a *Product Manager* - will have to interact with a multitude of teams, because the contributions of all teams are necessary to create a product. In such instances, while the *Product Managers* might be reasonably close to the clients, they will not be able to be as close to the teams as a typical *Product Owner*. In such settings, the role of a *Flow Manager* on the team side - who would be part of the team itself - would provide that necessary interface between the client and the team.

Takeaways

A *MOVE* is an essential component of the *TameFlow Approach*. It allows the coming together of many different perspectives - and in particular the economic perspective of *Throughput Accounting*; and, as we will see in the following chapters, it is essential in the structuring of the incoming *Work Load* in support of *Execution Management* and *Governance* in *PEST* environments.

Martin Nantel says...

My role in organizations leads me to work with several teams and make them perform. Visual work management is by far the avenue I prefer (kanban). In fact, the osmosis communication provided by our daily meetings as well as the visibility of the work to be done allows us to make continuous improvements to adjust ourselves in order to create more value for our customers.

At a certain moment or another, you observe problematic situations on your Kanban Board that you can not clearly identify, nor understand. You feel that something is wrong, but you are not able to put your finger on it. It's like the word you have on the tip of your tongue, but you can not say it.

You study and apply agile approaches, lean management, six sigma, business Analytics, change management and even more. You make improvements, but the results are usually disappointing, temporary. Through time, your readings, your reflections and your research, help you to put all the puzzle pieces together and then it's the illumination.

You finally found the answer to your questions.

The book "Tame your Work Flow" is exactly that. The answer to your questions.

It is the result of experience, professional maturity and the understanding of common problems encountered and how to mitigate them and innovate in a direction never explored before. The Theory of Constraints, Throughput Accounting, the Wait Time versus the Touch Time, the Drum Buffer Rope and many more approaches will take you to another level, a higher level, a level you have never reached before.

My friends do not look further, everything you need to know is in this book. Take all the time you need to read it and take ownership of its content. I guarantee you will not be disappointed. So much to learn from.

Steve and Daniel, thank you so much for this magnificent book. You guys rock!

Martin Nantel, Senior Business Analyst, Project Manager, Scrum Master, TameFlow CKP and LSSGB.

16—Introduction to Execution Management Signals

Most management approaches focus heavily on *planning* activities and then try to catch *deviations* from the plans to exercise risk management. Planning is a useful activity to gain insights and map out what is known from what is not known, and to prepare as best as possible for what is to come.

What most approaches fail at is not in the planning itself, but in how the *execution* is *managed*. In the *Tameflow Approach* we are not thinking about the *execution of the plan*, but more specifically about the *execution of the work* (work that was presumably planned) with the use of leading indicators and *Management by Exception*.

In environments full of **V**olatility, **U**ncertainty, **C**omplexity and **A**mbiguity (VUCA), trying to stick to a plan does not make sense. That's why Agile approaches were born, and knowledge discovery was put at the center of all activities (to create a "*learning organization*"), and governed through an iterative process with short cycles (like the *Sprints* in Scrum); wherein the governing mantra is to "*inspect and adapt*."

Agile approaches are not flawless, especially when it comes to high performing *organizations* (rather than simply *teams*). The main argument for this is that the *feedback loops* between what happens on the ground and the higher levels of management are too long and too slow.

Additionally, the information brought to top management is often *not timely enough* or *accurate enough* for them to support *operational decisions* that are *relevant, actionable* and *timely*.

In other words, Agile approaches, while being excellent at improving operations at the team level, do not include top management. In fact, it is almost a stereotypical truth that Agile approaches are seen as *anti-management*.

Note: Scrum even popularized this divide with the infamous "*Chickens and Pigs*" fable; and although it has now been retired from the official *Scrum Guide*, its folklore still persists, and it is often the hallmark of the sub-culture with which *Scrum* practitioners so proudly identify themselves with. Clearly, any approach that is divisive cannot foster the *Unity of Purpose* and the *Community of Trust* that must be at the center stage of high performing organizations.

Even established Agile approaches which are lauded as bringing about "business agility" are not very good at making it happen. Let us remember the simple definition of what "agility" is all about: being able to *change direction at high speed*. One key component to do that is to be *in control* while progressing at high speed.

The quote of the legendary race car driver Mario Andretti comes to mind here:

If everything seems under control, you're not going fast enough!

This has the obvious corollary that as long as you are in control you *can go faster*.

The problem here is that traditional *Agile approaches* do not provide control even while going slowly, and therefore cannot move to higher levels of performance.

Why?

Because the basic management-oriented feedback mechanisms built into Agile approaches are backward looking, not in real time. Rather, they only occur once the *Sprint* is finished, at retrospective time. Top management will be in the driver's seat, and has to move forward looking into proverbial the rear view mirror. No effective double loop learning is ever achievable with such limitations.

Agile approaches - and those who adopt them - fall victim to their lack of managerial pugnacity. Have they improved the bottom line in all these years of commercial success?

Of course, to their credit (sort of!), top managers are used to this, accustomed as they are to the *Cost Accounting* management reports. But there are much better ways; venues that give top management direct immediate control and the means for "changing direction at high speed."

Traditional Agile approaches have no such levers at their disposal and never will have.

For example, consider *Scrum*. The primary tools for action are the *Sprint Review* and the *Sprint Retrospective* ceremonies. The names themselves, "*Re-view*" and "*Retro-spective*," imply looking... well... backwards.

But to *change direction at high speed*, there needs to be something better. Something that provides instantaneous, almost real-time *feedback* on how things are going. And that this feedback is available at all the appropriate decision-making levels within the organization at the right time with the right content.

The feedback loop should not only be instantaneous but also *anticipatory* and should provide *leading signals of risk materialisation* about where to focus the next decisions. It should be a tool that allows looking *forward* rather than *backward* and allows us to see what is coming *toward* us. It should address proactively the *Work Flow*, the *Constraint*, the *Portfolio*, the *Work Process*, the *Work Load*, the *Scheduling* etc.

This is what we achieve with the *TameFlow Approach* when we start adding elements such as *Informational Flow* and *Psychological Flow* to the mix - elements which are absent from *Kanban*, *Scrum* (and their variants) or any other improvement approaches.

What is needed is a new nervous system in the organization that is able to detect and react quickly to any new information that is discovered while traversing the *VUCA* jungle. Conventional project management, executive management, business management and Agile approaches are simply too far detached from the nature of knowledge-work to make this happen.

Agile approaches, namely *Scrum*, are too narrow-minded, inward-looking into their team focus, and anchored in the belief that "scaling" is a solution. They mostly fail when dealing with large organizations exposed to *PEST* and *VUCA*.

In building this new *Tameflow* nervous system, one part is instilling into the organization the performance mindset to develop the attitudes that will make it work (*Psychological Flow* and *Mental Models*). Another part is literally *rewiring* the organization so that *Information Flow* moves in all directions, in the right format, in sufficient quantity, with only the relevant content, and at the right time.

With this kind of wiring, we will be equipped to lay out *Execution Management* and *Operational Governance* at the top organizational level with unprecedented performance and alacrity.

The Logic of Critical-Chain Project Management (CCPM)

One key element of this wiring comes from the *Execution Signals* of Dr. Goldratt's *Critical Chain Project Management* (CCPM). They support a dimension of *Flow* not found in pull systems like conventional *Kanban*.

Note that these signals come at *no cost*, since they are part of the *Work Flow* and *Work Process*, except for the managerial willingness to make them visible and actionable.

One of the most significant contributions of the *Theory of Constraints* to the field of project management is *Critical-Chain Project Management* (CCPM). Many practitioners of traditional Agile will want to run for cover just at the mention of the words “*project management*”. The words raise the fear that lots of unnecessary effort will be lost in up-front planning activities to produce Gantt chart-based schedules that in practice never materialize and that become obsolete almost immediately after a project has started.

Therefore, it would not be surprising that the same reaction is seen at the mere mention of *Critical-Chain Project Management*. Granted, CCPM is indeed used to plan projects in a conventional setting. Yet one of the major strengths of the approach is in terms of *managing the projects while they are executing with leading indicators*. This is what is needed to tame complexity: the execution management part of CCPM.

We want to apply the lessons of *Execution Management* that come from CCPM. We will not be concerned with the *Project Planning and Scheduling* aspects of CCPM.

But before we do, we need to understand the logic underlying CCPM.

The traditional way to plan dependencies in projects and to estimate their total duration is through the so-called *Critical Path Method* (CPM). It has been fraught with problems, starting with the estimates that the workers have to give, and for which they are then held accountable.

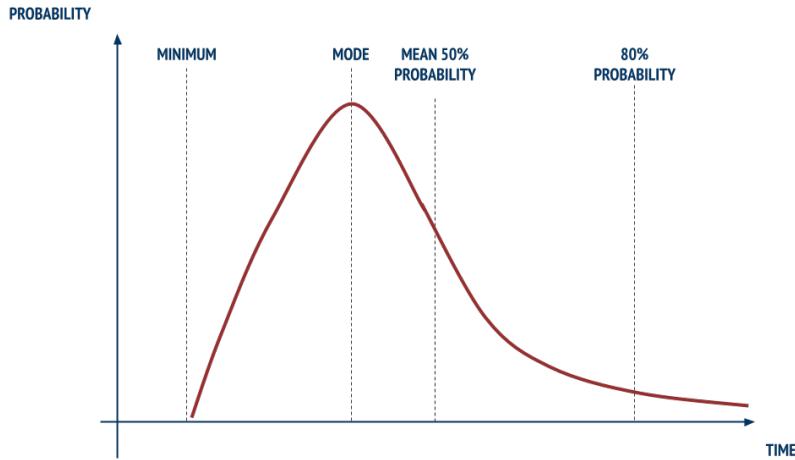
Typical psychologically driven behaviours that arise are:

- **Inflated Estimates:** They give estimates that are 2-3 times larger than necessary, typically to compensate for forced multi-tasking, and to have contingency “buffers” for things that inevitably “go wrong.”
- **Sandbagging:** Resistance to pass on any gained time to downstream activities, compounded with the avoidance of any additional *Work Load*.
- **Student’s Syndrome:** Cramming and leaving things to be done at the last moment.
- **Parkinson’s Law:** Work expands to fill the available time.
- **Gold plating:** Tendency to “polish” and “improve” completed work when finished ahead of schedule.
- **Handover Delays:** Reluctance or failure to hand over finished work.

The CCPM method was devised to address the above issues (as well as many others).

Point Estimates vs. Probability Distributions

A first significant insight is to accept that point estimates (that is precise numeric estimates) do not make much sense, especially in an overly complex world with ever increasing uncertainty. Project *durations* are better represented with *Probability Distributions*.



The distribution (area under the curve) represents the probability of a project being finished by the corresponding time on the X axis. For example, the *Mean* (50%) probability point denotes that the project has a 50/50 chance of finishing at that time. Likewise, the 80% probability point signifies that the project will finish by that date 8 times out of 10. Clearly, at some point in the future the project will have a 99.999% due date probability, and we can be virtually sure that it will be delivered. Note that the probability curve is “skewed” to the right, and it intuitively makes sense that it should be so.

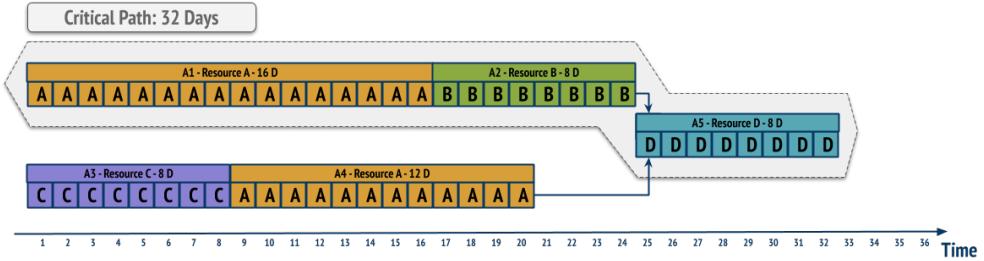
Critical Chain Planning

In CCPM there is the notion of a *Buffer*. It is not a device to compensate for “unforeseen” events. It is a device for monitoring variability and detecting risk materialisation in the execution of the project. Let us first understand how it is derived.

Let us consider a mini project, from a traditional project management perspective:

| ACTIVITY | RESOURCE | DURATION (Days) |
|----------|----------|-----------------|
| A1 | A | 16 |
| A2 | B | 8 |
| A3 | C | 8 |
| A4 | A | 12 |
| A5 | D | 8 |

The sequencing and dependencies of the activities are illustrated in the following diagram:



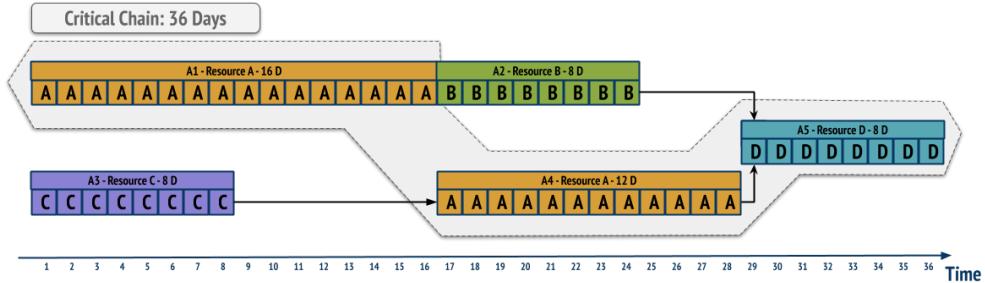
The dotted outline encompasses the *Critical Path* - which is the longest duration of all activities between the start and the end of the project. It also determines the shortest time possible to complete the project. The CPM method is primarily concerned about the dependencies between activities. The *Critical Path* is 32 days.

However, in the above figure we notice that resource A is doubly booked for eight days! We see that between day 9 and day 16, resource A should be working on activity A1 and A4 at the same time.

Critical-Chain Project Management (CCPM) will consider not only the dependencies between activities, but also which resources are available, and how their availability might change the scheduling of the project - something that is ignored by CPM.

So activity A4 needs to be moved forward in time, until resource A is available again (after completing activity A1).

The revised project schedule would look like:



The dotted outline encompasses the *Critical-Chain* of the project.

From a *Theory of Constraints* perspective, the dotted outline shows us the “constraint” of the project. It is what is limiting us from delivering sooner, and the goal of any project is to deliver sooner. In other words, the *Critical Chain* is the *Constraint* of the project - it is the shortest time required to complete it with the available resources.

The *Critical Chain* of the sample project is 36 days.

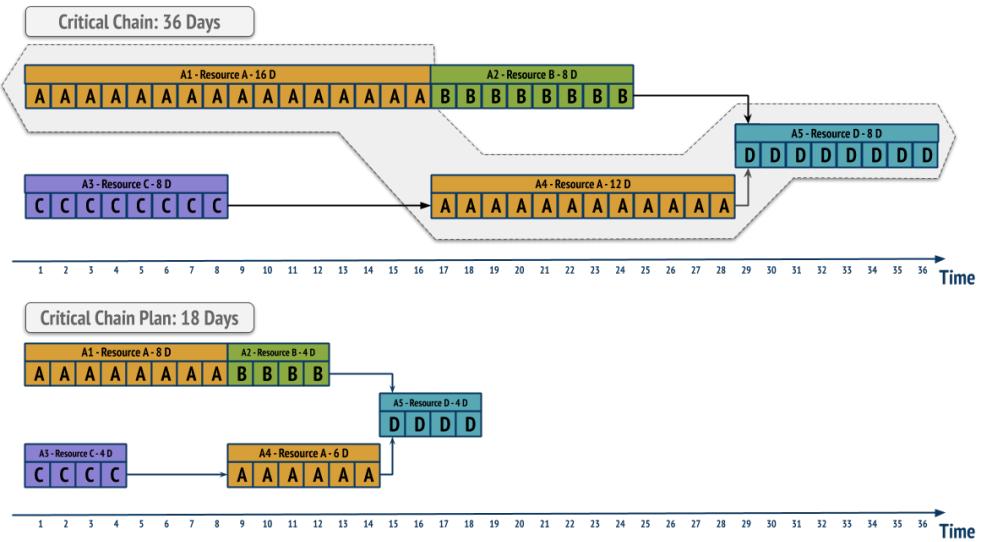
The Critical Chain Plan

Remember, we are looking at this from a project management planning perspective but with the aim of understanding how the execution management features of CCPM work.

The original plan is based on conventional estimates, and contains a lot of extra time (as observed above, with inflated estimates, sandbagging, Student's Syndrome and Parkinson's law).

What the CCPM project manager would do is to expose the “contingency” time hidden in each task and give it to the project.

To do this, we start by slashing all activity durations in half, and produce a *Critical Chain Plan* that is half as long as the original one.



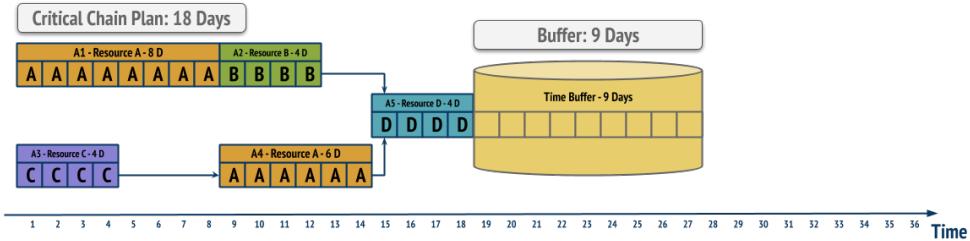
In the example, the *Critical Chain Plan* would be reduced to 18 days.

Stakeholders are made aware of the probabilistic nature of the project, and their expectations are set, communicated and managed in terms of due date probabilities, which are conceptually and visually captured by the *Buffer* and *Buffer Zones* (see following sections).

The Critical Chain Buffer

The next step in the CCPM planning would be to add a *Buffer* at the end of the project plan. The idea is to take the cumulative “contingency” time reserves that were taken out of every task, and give them to the project as a whole, as a *Project Buffer*. This *Buffer* is then sized to half of the cumulative duration that was saved from the tasks. In the example, that would result in a *Buffer* of 9 days.

Note: This sizing of the *Buffer* to half of the saved duration is a CCPM *heuristic* that works most of the time. In more advanced applications, the size of the *Buffer* can be set in other ways. What matters with respect to what we are learning now, is simply to understand why such a *Buffer* makes sense, and especially how it can be used for practical purposes. For situations where performance is critical, the sizing of the *Buffer* can be done with more sophisticated calculations.

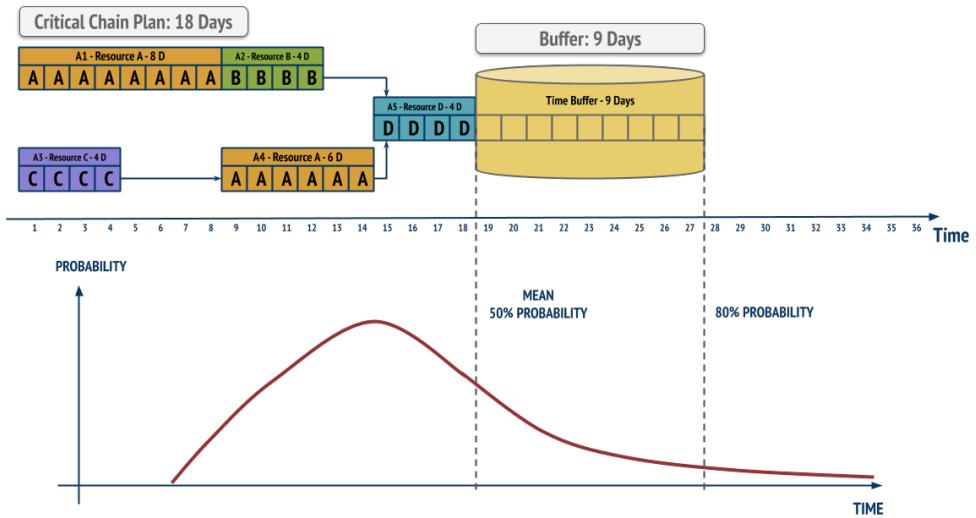


Now, what is interesting is when we superimpose the typical project distribution curve to the timeline of the *Critical Chain Plan*. We can imagine that the start of the *Buffer* coincides with the 50% due date probability (after all, we did cut the original estimates in half); and that the end of the *Buffer* corresponds - approximately - to the 80% due date probability.

How did we get to this 80%? Let's repeat the heuristic reasoning just so it is clear. We started with half of the original timeline; that is the 50% mark. Then we took half of that (25%) and added it to the end, so we arrive at $50\% + 25\% = 75\%$. Then we simply round up because the distribution is skewed and gets thin to the right, and assume we are happy to set that limit at 80%. It is a heuristic, a rule of thumb.

If (as we will propose later) we have historical *Flow Time* distributions as we do with our *TameFlow Boards*, then we wouldn't even need to do this. We simply take the 50th percentile and the 80th percentile to mark off the start and the end of the buffer.

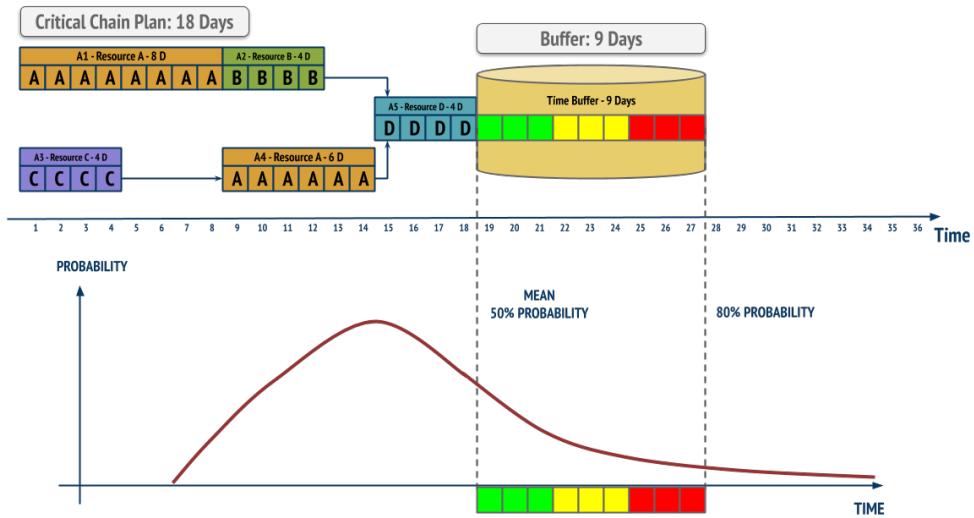
Note: Reasoning in terms of due date probabilities is something that could easily be transposed in defining *service level agreements* (SLA) to build trust with the clients on the basis of statistical data rather than promises.



Note: The manner in which the *Buffer* is placed and sized can become a science or an art in its own right. Here we are just exposing the most common heuristic to understand the underlying *reasoning*. We approach this as a *Mental Model*.

Buffer Zones

The next step is to imagine dividing this *Buffer* into three parts, the *Buffer Zones*, and conventionally color the first part in green, the second part in yellow and the third part in red.



Naturally, the three parts can be reflected on the due date distribution chart too.

Buffer Consumption and Buffer Burn Rate

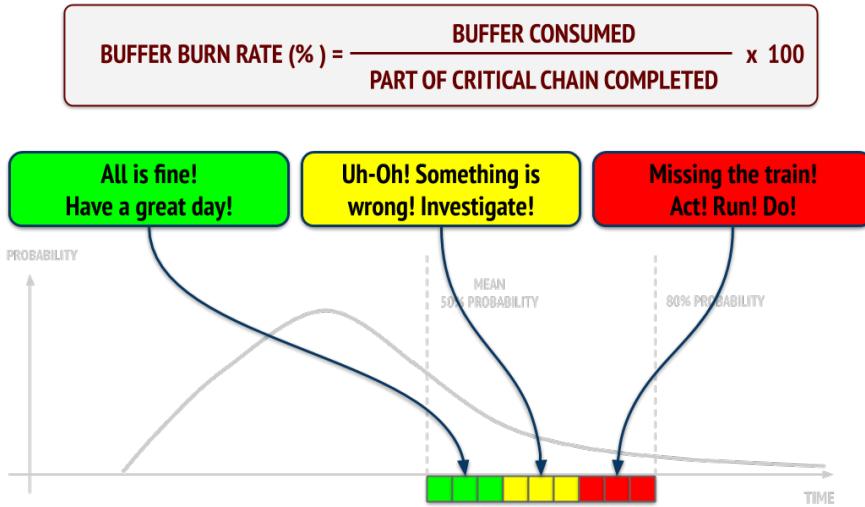
With this setup we can switch our attention from *planning* to *execution*. Once the project is being executed, we need to check how much is completed, and put it in relation to the *Buffer Zone* on our due date distribution curve.

Once we have the *Buffer* and its three zones, we compute the value of the so-called *Buffer Burn Rate*. The *Buffer Burn Rate* is the ratio between the amount of *Buffer* consumed and the actual work performed on the critical chain. The *Buffer Consumption* can be thought of as the accumulation of activity “lateness” on the critical chain.

Note: We may use the word *Buffer* here, but it has to be understood with respect to context. Unlike “buffers” that we find elsewhere and which are effectively “padding”, we do not pay for the *Buffer* by trading money to reduce uncertainty. The word “buffer” is not quite right as the CCPM *Buffer* is in fact an operational monitoring device that gives significant leading signals of oncoming risk materialization at no cost. Hence, a CCPM *Buffer* is not there to *reduce* variability. Its purpose is to *detect* unfavorable variability early enough so that we can act and keep project duration under control.

Execution Management Signals

While we might at first be tempted to report the “color” of the *Buffer*, and give a “status report” on how a project is going, what matters more are the thresholds between the three *Buffer Zones*. When the *Buffer Consumption* crosses the thresholds, we can have leading and actionable signals that can direct our attention and decision-making.



The rules are simple:

- As long as the *Buffer* is green, everything is “normal” and we have nothing to worry about.
- As soon as the *Buffer* switches from green to yellow, we need to interpret that as a signal to start looking into reasons for potential lateness. The project is in no way challenged, but it is starting to accumulate lateness. It is an indicator of plausible *Common Cause Variation* (which the *Kanban Method* ignores to manage).
- As soon as the *Buffer* switches from yellow to red, we must do something because if the trend continues, the project will be late. It is an indicator of plausible *Special Cause Variation*. (The *Kanban Method* labels all variation as SCV, without distinction.)

Note: A red *Buffer* does *not* mean that the project *is* late, but that it is *running a high risk of becoming late*. Hence, this is an early warning system that gives us leading signals of oncoming problems. And

we have time to react and fix or mitigate any problems we might identify.

The model is visual, scientific and quantitative. It also puts all projects on relative scales so it is easy to compare the status of one project to the next. Project size does not matter as all projects are normalized to a common scale (a ratio between percentages).

We can thus perform comparative analyses of multiple heterogeneous projects, despite huge variability in project scope, budget, size, schedule and even different exposure to variability at the glance of an eye!

Note: Given the logic of how the *Buffer* produces signals, it is clear that the sizing and placement of the *Buffer* is a critical element. A *Buffer* that is too big will give signals that are too late. A *Buffer* that is too small will give many false positives. But for the moment what matters is understanding what the *Buffer* is, how it works and why it is useful.

Visual Execution Management and *MOVEs*

At the end of *Chapter 15 - Outcomes, Values, and Efforts in PEST Environments* we observed that in the *TameFlow* simulation we could consider each project as a *MOVE*. In fact, this is what we will do in general. Any work that will be handed out to a team will be packaged in *MOVEs*.

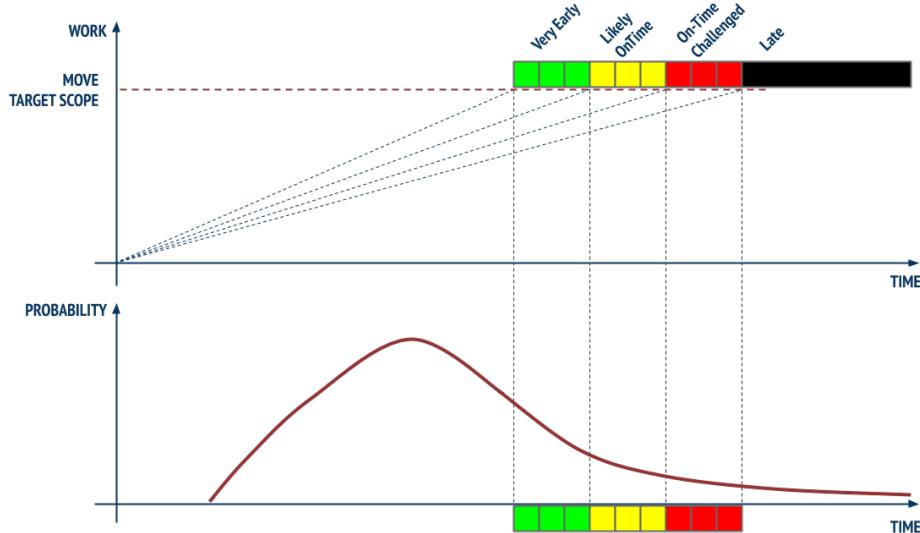
The smaller the *MOVE*, the better (because we know that smaller batch sizes are beneficial). But we cannot escape the logic of having to deliver some *Business Outcome* and/or *Value* with *Minimal Effort*.

Note: This has far-reaching consequences for those organizations that hand out story or feature requests to their team on someone's whim. If that story or feature does not bear a sound business outcome or throughput value, it should be refuted; and maybe repackaged into something bigger (the *MOVE*) that does indeed deliver outcomes and values. The requirement of delivering qualifiable outcome and/or quantifiable value is an excellent way to proactively take care of *demand shaping*, and not just start working on things reactively.

The *MOVE Buffer*

We can observe that the timeline used in showing the due date probability distribution is effectively the same as the one we have used in the *Work/Time* diagram that we became familiar with from the very first chapter. If we put this in relation to the *Target-Scope* of the *MOVE*, and visualize the *Buffer Zone* onto the

Target-Scope, we can recognize that the *Target Scope* does not have a fixed delivery date, but a range of dates, based on probabilities.



We can thus project the *Buffer* on either diagram, and we can reflect on the color coding with respect to the actual delivery of the *Target Scope*:

- **Green:** very early delivery
- **Yellow:** on time and likely delivery
- **Red:** challenged but still on time delivery
- **Black:** delivery is definitively late

To distinguish this *Buffer* from the CCPM *Buffer* we will refer to it as the “*MOVE Buffer*.”

Furthermore, the *MOVE Buffer* is also distinct from the *Buffer* we encountered in *Chapter 14 - Flow Efficiency, DBR and TameFlow Kanban Boards*, which we called the “*DBR Buffer*.”

Both of these *Buffers* are derived from the *Theory of Constraints*. The *DBR Buffer* had its first applications in inventory management in manufacturing. The *MOVE Buffer* - as we just saw above - derives from the *CCPM Buffer* which is part of *Critical-Chain Project Management*.

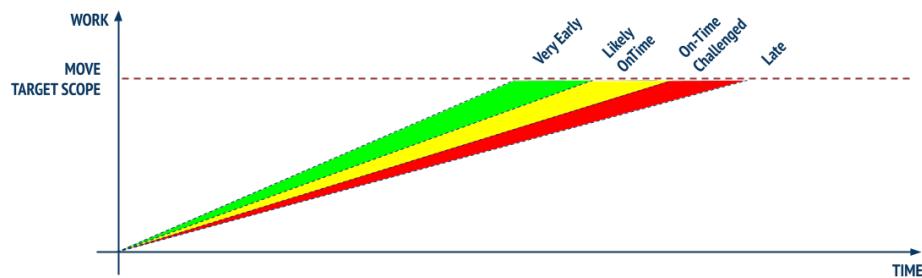
Note: In the *TameFlow Approach* we use *both* of these *Buffers* - the *DBR Buffer* and the *MOVE Buffer* - *together* and at the *same time* to manage knowledge-work, which is a unique proposition in modern management. By considering *knowledge-work* as *Flow*, we have elements of *Operations Management*. Hence the idea of a material *Value Stream* where the *DBR Buffer* plays the role of protecting the *Constraint* in the *Work Process* from starving. At the same time, like in *Project Management*, we have much more variability and need to hit deadlines. Hence the idea of protecting the deadline - or more

precisely the *Critical Chain* of the knowledge-work processes - through the *MOVE Buffer*.

Visual Execution Management

The *Buffer Management* techniques we find in *CCPM* are most valuable - and the only part that we will take from *CCPM*. They allow for sharp, crisp, instantaneous and brilliant *Visual Execution Management* that allow us to maintain control while proceeding at high speed and benefiting from leading indicators of oncoming problems. Along with the *WIP Ageing* signals, it is the only leading indicator of unfavorable variation of a *Work Process*, irrespective if it is under *Throughput Management* with the more advanced *Tameflow kanban boards*

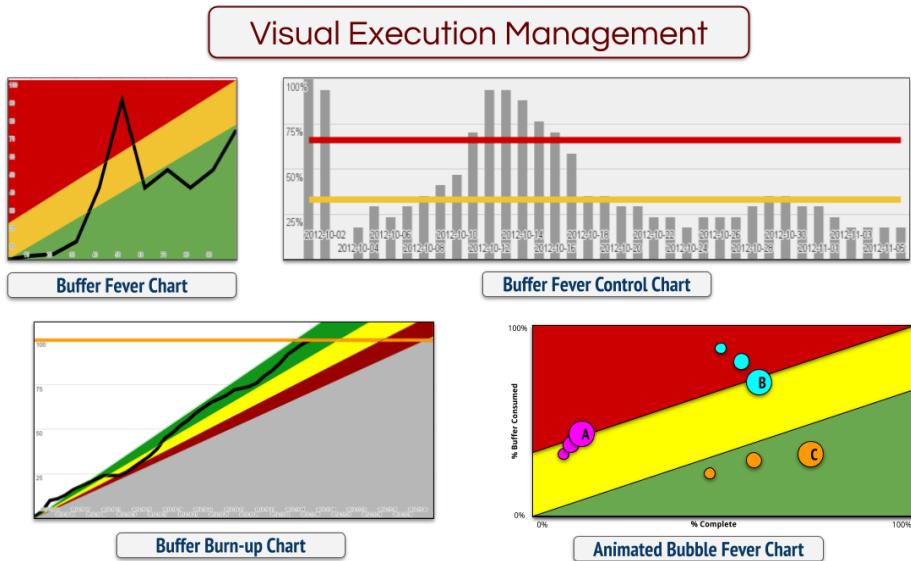
For example, if we are using the *Work/Time* diagram as a *MOVE* is progressing, (in other words, we are plotting a *Burn-up Chart* or a *Cumulative Flow Diagram*), we can represent the three *Buffer Zones* visually on the chart, like this:



Note: The “fanning” shape illustrated above should not make us believe this is some sort of depiction of the *Cone of Uncertainty*, that is so often used in project management theory. It represents a (sub-)range of due date probabilities.

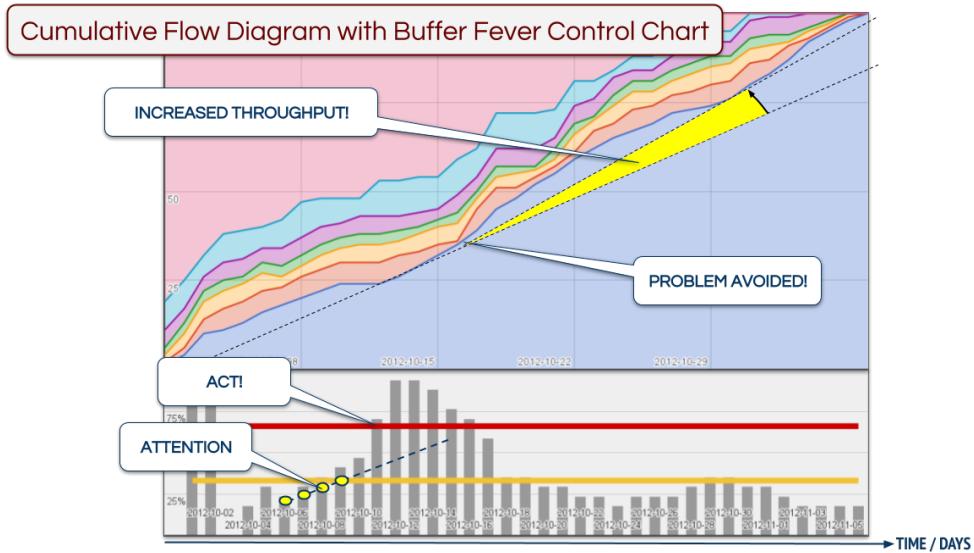
There are many ways in which the *Buffer Burn Rate* can be visualized. In the illustrations below you can see the most common ones, in addition to the *Burn Up* chart. We will explore some of them further in *Chapter 19 - Execution Management in PEST Environments*.

- The *Buffer Fever Chart*: This is the classical visualization that was first introduced in *Critical Chain Project Management*. It plots directly the percentage of work done compared to the percentage of *Buffer Consumption*.
- The *Buffer Fever Control Chart*: Shows the *Buffer Consumption* as a bar chart representing the *Buffer Consumption* on a day by day basis. The time axis is the same as the one we have seen in this chapter for the due date probability distribution chart and for the *Burn-up Chart*.
- The (*Animated*) *Bubble Fever Chart*: highlights the relative performance of multiple *MOVEs* in a single illustration. More about this in the next few chapters.



Note: The colors in the *Buffer* charts and the *Burn-up* chart might seem to be reversed, but they are consistent. The reversal comes from the fact that in the *Burn-up* chart we go from green to red as time progresses (on the horizontal axis), while in the other charts we go from green to red as *Buffer Consumption* increases (on the vertical axis). Also another detail to be aware of is that what is being plotted on the *Buffer Burn-Up Chart* is naturally the line showing the amount of work delivered, while in the other charts we visualize the *Buffer Burn Rate*. They are not the same, but they provide similar *Execution Management Signals*.

Operationally, a very convenient and practical aspect is that a *Cumulative Flow Diagram* (CFD), a *Burn Up Chart* and the *Buffer Fever Control Chart* share the same time axis. Putting the *Buffer Fever Control Chart* right below a CFD is an advanced visualisation technique that has no equal in the Agile and Kanban worlds - as you can see in the following illustration:



It is evident that this technique is easily leveraged with the data that your basic kanban system already possesses. The average *Flow Time* of each *Work Item* can be used as the 50% cutoff; add half of that and you have the end point of the *Buffer*. This is a great leading indicator at no cost.

The (animated) *Bubble Fever Chart* is an advanced visualization that will allow us to have an overview of the execution of multiple projects. It will be presented in more detail in *Chapter 20 - Operational Governance in PEST Environments*.

From One Team and One *MOVE* to One Team and Many *MOVEs*

One of the premises of *Flow* is to reduce multitasking as it is the number one source of *Wait Time*. This means that we will strive to execute one and only one *MOVE* at a time. We will structure the whole planning, execution management and governance of our *PEST* environment to avoid this circumstance.

Consequently, if a team has to work on two or more *MOVEs* - and they will certainly have to in a *PEST* environment - those *MOVEs* need to be scheduled sequentially, with no overlap. We are building this on the premise that more business value can be achieved via speed and flow efficiency rather than on high utilization and resource efficiency. We do not mind if *workers wait for work*, but we will be hypersensitive and intolerant to any kind of multitasking and parallelism.

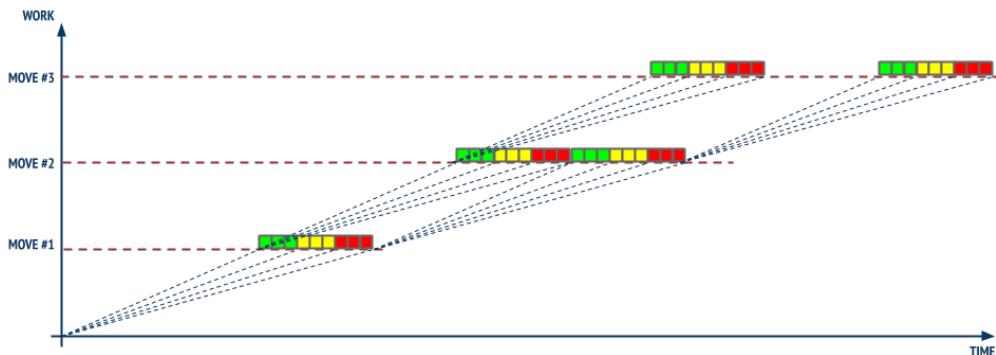
In practice, teams will work on one *MOVE* after another. As the delivery due date of one *MOVE* cannot be a fixed date but a range of due date probabilities, the effect is that the *Start Date* of the following

project is also be set variably by that range of due date probabilities of the earlier project(s). We will have the firm intention instilled as a cultural attitude, as soon as work on one *MOVE* is done, to immediately move on to the next *MOVE*. (See why we call it a “*MOVE!*”) We will actively counter the common diseases like sandbagging, Student’s Syndrome, Parkinson’s Law.

Note: This does not mean that the entire system will be working on *only* one *MOVE* at a time. The important criteria is actually that the *Constraint* is not *Multitasking* and that it is working sequentially from one *MOVE* to the next. As we learned in *Chapter 14 - Flow Efficiency, DBR and TameFlow Kanban Boards*, if we use *DBR Scheduling*, then we will certainly have instances where the *Constraint* is working on one *MOVE* while the upstream teams are working on the next *MOVEs* with the deliberate objective of keeping the *Constraint* busy. (We effectively start “staggering” *MOVEs* according to the drum beat.) So the system might be working on different *MOVEs* at the same time, though any single team/resource will work on one and only one *MOVE* at a time.

In our *Work/Time* diagrams, the *Target Scopes* of the subsequent diagrams can easily be stacked one on top of the other on the vertical axis. The start “time” will happen in the sort of accordion we have at the end of each *MOVE*, representing its due date probability range.

If we had three *MOVEs*, it could look like this:



In the above illustration we have kept the same size for three target scopes - for ease of illustration. In the real world, the scopes would be different. For illustration purposes, we have extrapolated the “greenest” and the “reddest” due date delivery dates from one *MOVE* to the next. In the real world, the

actual start of the next *MOVE* will most likely be somewhere in between, and at times - when things go sour - they will be beyond the red zone and in the black zone of the previous project's late delivery.

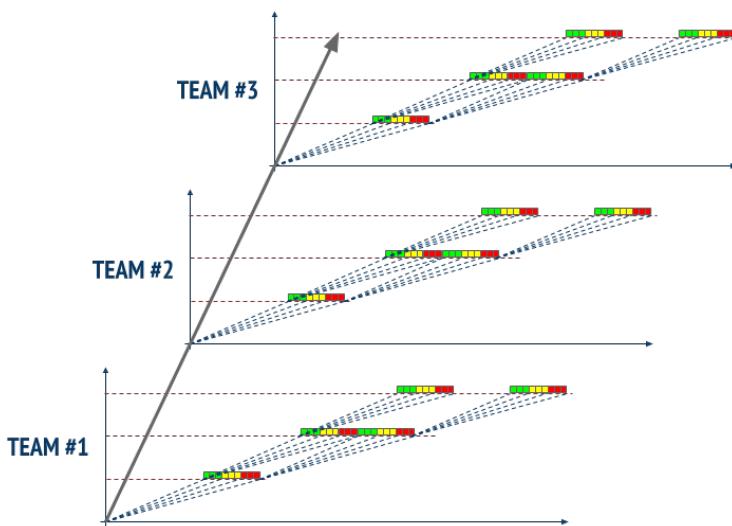
Notice that unlike most Agile approaches, and Scrum in particular - with the *TameFlow Approach* - variability is managed on the *time* axis rather than on the *scope* axis. This is actually a huge advantage compared to Agile methods. By taking the liberty of varying scope - as is customary in Agile - it is too easy to *add even more variability as a result of going back and forth with scope management*. The time dimension, on the other hand, unrolls only in one direction, from the past to the future, and what is gone is gone. With lesser degrees of freedom, the *Tameflow Approach* gains in simplicity - as Dr. Goldratt taught.

From One Team and Many *MOVEs* to Many Teams with Many *MOVEs*

The above illustration showed how many projects are determining a sort of “folding fan” of due date probabilities, of one *MOVE* after the other. Of course, as we progress on the *MOVE* currently being worked on, the fan of the following projects will get narrower. Once the *MOVE* is done, we know we can start the next one, and at least its fan will “converge” to a fixed starting point. The whole effect naturally cascades to the subsequent *MOVEs*, almost in a fractal manner.

This situation is not much different when dealing with several teams. We can simply imagine the same kind of *Work/Time* diagram piling up in the third dimension, with one such diagram for each team.

We can imagine it as follows:



This is a simplified visualization with many compromises; but it gives us the right *Mental Model* to further reason about *PEST* environments.

Imagine scaling this to a point of purview covering some 120 teams and 70,000 *MOVEs* per month, as was the case mentioned in the *Introduction*. That is the kind of scale that we can manage by adopting the appropriate *Mental Models*.

Note: The software engineering industry is very much concerned about “*scaling*” Agile processes. It is the authors’ opinion that traditional Agile - alone - does not scale. A better option is to scale our *Mental Models* as to be able to handle larger domains. To the traditional Agile practitioners: don’t be concerned that we are in any way losing sight of the value of human interactions, personal development, social constructs and other “soft values” that are very much associated with mainstream Agile. That is all covered in the *TameFlow Approach* by explicitly addressing the *Psychological Flow* in ways that actually go beyond what Agile ever achieved.

Takeaways

We have covered the basics of CCPM at the beginning of this chapter, but one thing must be made clear: we only retain the idea of the *Project Buffer* from the CCPM approach, and use it to produce *Execution Management Signals*.

By combining Kanban with the *Buffer Management* techniques of *Critical Chain Project Management*, we achieve explicit risk management. Explicit risk management has no other point of reference in Agile - nor the *Kanban Method* - and is one of the biggest contributions of the *Tameflow Approach* to knowledge-work management.

The origin of *TameFlow's Explicit Risk Management* date as far back as 2012, when Steve Tendon prepared a series of blog posts for the Lean Kanban Netherlands 2012 conference, the topics of which were later included in the “*Hyper-Productive Knowledge Work Performance*” book.

Understanding these topics is crucial in the mastery of Agile risk management. Why? For the single reason that with *Explicit Risk Management*, not only can we act on leading indicators and cater for problems before they become critical; but we are also able to identify *Common Cause Variation*. Identifying *Common Cause Variation* is the cornerstone of lasting ongoing improvements. But in order to detect *Common Causes*, we must have a game-plan centered around the accompanying *Signals* that trigger us to investigate them. These *Signals* come from setting up and using *Buffers* - both the *DBR Buffers* and the *MOVE Buffers* - as taught by Dr. Goldratt with CCPM.

Michael Küsters says...

Having been involved in Organizational Change Management and Agile Transitions for a long time, it's utterly refreshing to see such a profound source of inspiration.

The authors have connected critical bits and pieces with formerly missing links to go far beyond where typical "Agile Approaches" would take us.

This book provides a clear way to break past the flawed assumptions and mental barriers inherent to many of today's "Agile" framework implementations.

Michael Küsters, CEO of Intelygence GmbH

17—Introduction to *Full-Kitting*

One of the most important teachings of the *TameFlow Approach* is that once the organization begins working on a piece of work (i.e. when *Touch Time* starts ticking for the first time), the goal is to never interrupt work on that piece until it is completely finished and delivered. The *Flow* from start to finish should be unhindered.

This is an ideal that will not always happen in the real world. However, a great deal of “stop and go” and “hiccups” that we observe are actually self-inflicted. In fact, in the first few chapters of this book, we discovered how the majority of *Wait Time* is self-imposed.

Introduction to *Full-Kitting*

We can thus strive to pre-empt as many stoppage causes as possible *before* work is started. This activity is known as “*Full-Kitting*.”

The notion stems from the manufacturing world (like many ideas we appreciate in *Constraints Management*), where before work is started, the most productive organizations will ensure that the “*Full-Kit*” is available, such as all the right raw materials, tools, information, workers with all the necessary skills, and so on.

We can adopt a similar stance in knowledge-work and avoid being caught by surprise in the middle of a work process only to discover that *something* was missing. As we know, it is better that work not start at all and stay in Disneyland queue time in front of the *Work Flow*, rather than being stopped *inside* a *Work Process* in the *Work Flow* and incurring *Wait Time*. This gives us options, and ensures stable and predictable work flow.

We don't have time for this!

One objection that immediately needs to be addressed is that “we *don't have time for this and we must focus on actually working instead*.” Or that starting with imperfect information is a sound *Mental Model*.

The objection has no foundation, because typically any *Full-Kitting* activity will be performed further upstream and outside of the *Work Flow*, and in particular, we can organize ourselves in such a way that the *Constraint* is not (or very rarely) involved in the *Full-Kitting* activities.

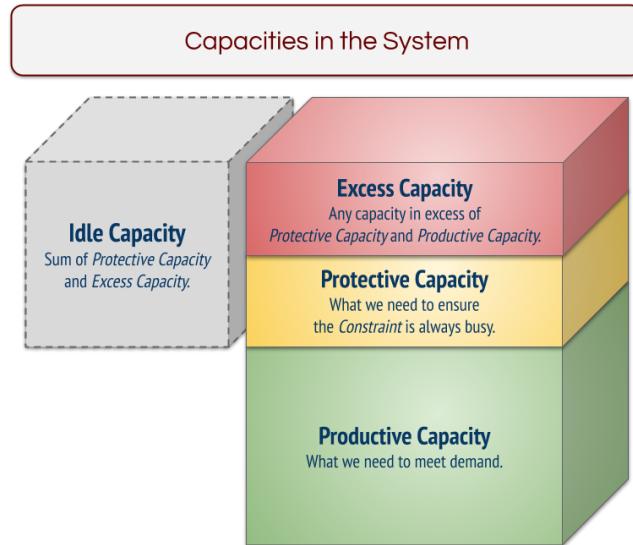
There are at least three explanations as to why this objection is unjustified:

- We forego the opportunity of capitalizing on *Excess Capacity*.
- We forfeit seizing a zero-cost benefit.
- We miss the opportunity to improve the *Work Process* (“sharpening the saw”).

Let us see what this means.

Capitalizing on Excess Capacity

We need to be more sophisticated on our mastery of *Capacity* simply by taking the *TOC* perspective, and distinguishing between the following:



“ Productive capacity: The capacity required to operate. It can be viewed as the “delivery” line that we have seen in all *Work/Time* diagrams in the first chapters.

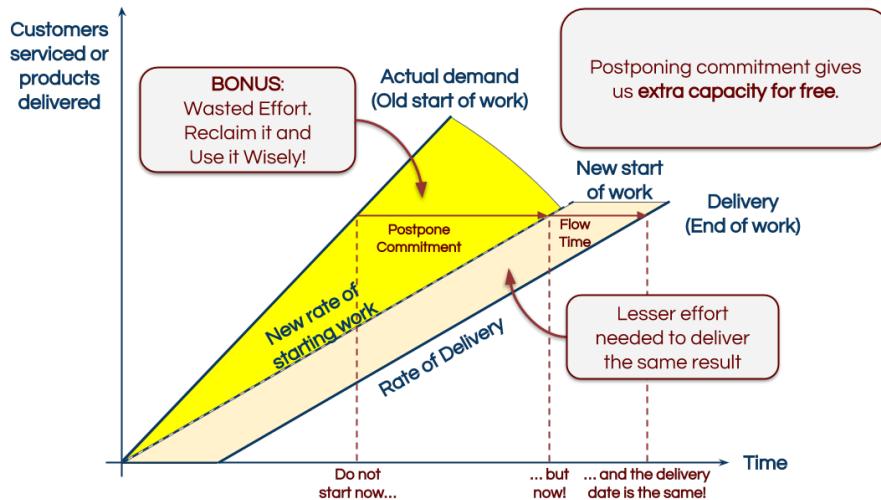
“ Protective capacity: The additional capacity needed on the *Non-Constraint* resources to protect the *Throughput* of the system. When disruptions occur during *Work Execution* of the *Non-Constraint* resources, the *Protective Capacity* is what allows them to “catch up” with the *Constraint* without forcing the *Constraint* to slow down. This capacity allows the system to absorb small hiccups without affecting the overall performance and still keeping the *Constraint* busy.

“ Excess capacity: All capacity that is not used either to produce or protect *Throughput*.

“ Idle Capacity: The sum of *Protective Capacity* and *Excess Capacity*.

Note: Once we become aware of these different categories of *Capacities*, we must avoid falling into the *Cost Accounting* trap. The *Cost Accounting* mindset will impulsively want to get rid off any “idle” or “excess” *Capacity* - to save costs; but this would be a mistake as we will see in a moment. This is where the *Mental Model of Cost Accounting* needs to be replaced with the one of *Throughput Accounting*. Otherwise all the benefits of flow would be in jeopardy before even starting.

If we revisit the diagram we used while learning about the value of postponing commitment (in chapter 2), we can understand that the area highlighted in yellow, which we called “*Wasted Effort*” is actually representative of the *Idle Capacity* which we can break down in *Protective Capacity* and *Excess Capacity*. When implementing the techniques for limiting *Work In Process* (via *DBR Scheduling* and focusing on avoiding *Multitasking*), we will be able to free the *Excess Capacity* and use it wisely.



Note: The yellow area in the diagram above represents effort that has actually been spent - not *Capacity* that is available for future work. Most of that *Capacity* was probably drained away down the *Multitasking* hatch. It is therefore not possible to discern between *Protective Capacity* and *Excess Capacity* directly on the diagram, as they are context dependent with the *Work Load* distribution on specific points in time, in relation to the actual resources available at those times. The point is to *Postpone Commitment* so that such *Capacity* is *not* spent in unproductive efforts at all, and hence becomes available .

Clearly some of that *Idle Capacity* has to be reserved as *Protective Capacity* so that the *Non-Constraint* resources can “protect” (subordinate to) the *Constraint* and ensure there is always enough work to keep the *Constraint* busy. This is the kind of *Capacity* that is necessary - for instance - to keep the *Buffer* well replenished in any sort of *DBR Scheduling*. A system lacking *Idle Capacity* will need to compensate with higher *WIP* and constant fire-fighting.

What we have in addition to the *Protective Capacity* is the *Excess Capacity* that we can now utilize for better purposes.

Note: In the *Cost Accounting* world, all *Idle Capacity* is considered as waste that should be trimmed and become the target of cost reduction exercises. The reason why cost reduction exercises often produce disastrous effects is that they inadvertently chop away even the *Protective Capacity* with snowballing effects on performance, and creating the *Moving Constraint* syndrome. Often we recognize that this is happening when the company goes to the extreme of firing “redundant” employees.

A more performance-focused mindset would instead value such *Excess Capacity*. One way is to use it as “slack” (as the Agilists propose), i.e. as time that can be used for staff to learn or do other complementary activities such as helping to increase staff liquidity; re-engineering; fixing bugs; participate in *Full-Kitting* activities, etc. Another way is to *preemptively* prepare the forthcoming work, so that it can flow through the system more swiftly. *Full-Kitting* is instrumental for *Flow*.

Note: Managing *Protective Capacity* (finding the tradeoff with *Excess Capacity*) can also have far reaching consequences. By increasing *Protective Capacity*, the buffers can be made smaller (because in general less time would be needed to replenish them), and due-date performance would also increase. A welcome effect of increasing *Protective Capacity* is that less *Work in Process* is needed upstream of the *Constraint*. Notice the paradox here. Explicit *Buffers* might trigger the management reflex of making them bigger in the belief that the system will produce more *Throughput*. That would increase the *Work in Process* in front of the *Constraint* and as we know higher amounts of overall *Work in Process* actually *reduce* overall *Throughput* - one of the most counter intuitive management quagmires ever!

Seizing a Zero-Cost Benefit

In the illustration above we referred to the yellow extra capacity area as a bonus. The logic is that - as we know from *Throughput Accounting* - any extra work performed by any *Non-Constraint* incurs zero cost for the business, because the employment of any *Non-Constraint* is already accounted for under *Operating Expenses*.

Doing the kind of work that is required for *Full-Kitting* with the *Excess Capacity* of the *Non-Constraint* resources is something that can be *done for free*.

The objection that salaries have to be paid does not hold because those salaries are already accounted for in the *Operating Expenses* component of *Throughput Accounting*. Remember that *Throughput Accounting* deals with the differential impact of our decisions. Activity or idleness of *Non-Constraint* resources are

not going to cost extra money. It is money that would *necessarily* have to be spent even if those resources remained totally idle. If they do more work, we don't spend more. If they do less work, we don't spend less. We just need to employ that *Capacity* in ways that don't sabotage the realization of the business *Goal* - as all *Cost Accounting* shops consistently do!

The *Excess Capacity* comes from the fact that the *Non-Constraint* resources have more capacity than the *Constraint*. They should *really* remain *idle* most of the time, and at least spare some reasonable *Protective Capacity* for the reasons explained earlier (for the *Subordination* step of the 5FS).

Note: Remember that in high-performing organizations, *all workers* (except the *Constraint*) *have to wait for work*. Instead, the norm in organizations is that work has to wait for workers (accumulating *Wait Time* in all sorts of queues.)

Since we have *Excess Capacity* available, the *Non-Constraint* resources should remain idle as to not overload the *Constraint*; so we might as well utilize that *Capacity* to improve the performance of upcoming *Work Execution*.

Improving the Work Process

Excess Capacity can also be used to improve the individual *Work Process* involved. But we must be careful to aim the effort at the *Constraint* in the *Work Process* - because we know that improving anything else is a total waste.

It might prove difficult though, because - by definition - the *Excess Capacity* belongs to *Non-Constraints*. So it is like someone who is *not* doing the actual work intervenes to improve it - all this while the *Constraint* is actually busy doing its work.

Luckily, there is an easier approach. Simply use the *Excess Capacity* to *help* the *Constraint* directly. Even though this requires skill liquidity, it could eventually result in direct improvements of the *Constraint's Work Process*.

In fact, any activities made possible with the *Excess Capacity* that help ensure the *Constraint* is optimally employed, or that other *Non-Constraint* resources can offload work from the *Constraint*, or elevates the capacity of the *Constraint* itself - are clearly aligned with the overall philosophy and spirit of *Constraints Management*.

We Cannot Do Big Up-Front Design in a VUCA World and be Agile

Another objection that typically stems from proponents of Agile approaches is that they mistakenly believe that *Full-Kitting* is akin to "*Big Up-Front Design*" and would make the whole process fall under the dreadful "*waterfall*" model.

Agile likes to forge ahead with imperfect information. The problem is not in starting off with imperfect information, because that is a necessity in a VUCA world, and the *TameFlow Approach* is perfectly equipped to handle the situation. The problem is in forging ahead, and then handling imperfect information once its impact becomes evident. Typically, lack of information will manifest itself as "blockers" and/or "flowbacks"

- both of which interrupt *Flow*. Most of the times in ways that can actually be preempted with a little *forethought* and *coordination*.

Some will object that *Full-Kitting* smells of *Big-Up-Front-Design* because it would impose to design a solution before even the teams have understood the problem, and preventing emergent design. That it is a phase gated approach all over again, and that collaboration and co-creation is replaced by a formal sign-off of a written requirement artifact, that is treated as a hand-over, and that detailed designs are preferred over conversations.

While *Full-Kitting* might include design work, it is certainly not big nor is it up-front either. It would typically be just right-sized, and more likely scheduled to be ready just-in-time for the actual work to be released into the *Work Flow*.

Full-Kitting includes so much more than just design activities: it also encompasses cross-functional and cross-team knowledge discovery work. It is therefore paramount to understand how *Full-Kitting* fits into the overall picture of how we want to manage value creation in a *PEST* infected *VUCA* world.

Its primary purpose is to establish and ensure the conditions of *One-Piece-Flow*. It is similar to how surgeon theaters are checklisted before operations start. Or pilots use checklists to prepare their aircraft for a specific maneuver (like take-off, landing, etc.).

Full-kitting is an attempt at being mindful not to interrupt *Flow* once work has started, by *being prepared*. And if imperfect information is a necessary evil, then in the *TameFlow Approach* we deal with it by swarming onto the *Work Items* rather than blocking or doing a *Flowback*. It is in the swarming that knowledge discovery and emergent design happen. The difference is that there is a deliberate preparatory phase, the intent of which is to prevent interruption of *Flow* as much as possible, because the business value of *Flow* is so precious.

Note: *Full-Kitting* must not be confused with *Backlog Refinement* or *Backlog Grooming*, nor with the *Definition-of-Ready*, which are common in Agile parlance. Although *Full-Kitting* will also encompass similar activities and concepts, the major focus of *Full-Kitting* is to create the mindfulness, across the entire organization, toward not interrupting the *Flow* of work. And also - as we will see in a later chapter - to involve top management in critical moments of the decision-making processes.

Dedicated Roles

The *Full-Kitting* activity is so important that it should actually take center stage on how the entire *portfolio* of projects is managed. Since the activity is ongoing and essential, some organizations are advised to institute an official role that could be called the *Flow Manager* (or alternatively the *Full-Kit Manager*).

While instituting a dedicated role could be something to consider in larger organizations, the important notion here is that the *Full-Kitting* activities are performed.

In an Agile setting, the role that most likely would perform the *Full-Kitting* activities would be the *Product Owner*. In more traditional or larger organizations it could be a *Product Manager*, *Project Manager* or maybe a (project or product) *Portfolio Manager*. Though none of these existing roles have the specific purpose of a *Flow Manager* or *Full-Kit Manager*.

In some instances of the *Kanban Method* the roles of *Service Delivery Manager* and *Service Request Manager* are described, and they perform somewhat similar functions.

Regardless of the official - or unofficial role - what matters is that the concept of *Full-Kitting* be understood and the activities of *Full-Kitting* be performed.

The Nature of *Full-Kitting*

While *Full-Kitting* has the primary purpose of facilitating uninterrupted *Operational Flow*, it actually pertains more to *Informational Flow*. The predominant activity of *Full-Kitting* encompasses the gathering, coordinating, and disseminating of information.

It also has elements of *Psychological Flow* because one of the effects of properly performed *Full-Kitting* is to get the various players onto the proverbial “same page,” with a shared understanding of what concerns are being addressed; what trade-offs are being made; what is important and what is not. In that sense, it is very Agile.

The individuals put in charge of the *Full-Kitting* activity must have excellent communication and diplomatic skills, as they will stand at the intersection of many opposing demands and forces. They will be confronted with tension and friction; the crossway between the external world of clients, competitors and regulators; and the internal world of limited resources, roles, skills and, of course, power plays.

They are placed at another critical organizational interchange - the interchange between the value generating part, and the value ideation and design sector of the organization. They can have a huge impact on giving *shape and form* to the incoming work. In fact, jockeying is one of their primary responsibilities, because all the forces they must contend and interact with - at the end - will be represented in the *work* they decide to put into the pipeline.

How can they do this in a sensible way, and keep everyone happy?

Simulating *Full-Kitting* on the Client-Side

Back to the simulation game. In the earlier rounds, especially when all ten projects were being handled simultaneously, it was very likely that the *Product Owners* experienced a lot of confusion when figuring out which coins should be handed over to which front-line team.

Likewise, the teams would start to face greater and greater “housekeeping” issues, in order to not mix up the coins of one project with those of another. Finally, the integration phase performed by Team D had to put all things together into one deliverable that the *Product Owner* would then hand over to the *Customer*.

The participants are invited to reflect on the fact that when the simulation was run in the orderly fashion (according to the prioritization and sequence ranking of the projects), the *Product Owner* would experience a lot of idle time - simply because they were waiting for their turn in line.

It makes sense for the *Product Owners* to utilize that idle time to prepare the project work so that it can flow better through the delivery organization. This is where the *Product Owners* can simulate *Client-Side Full-Kitting*.

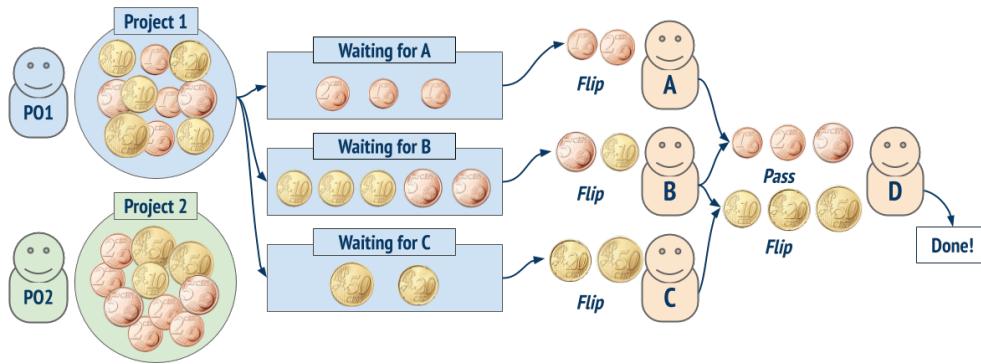
We are talking about “client-side” because this kind of *Full-Kitting* is performed upstream of the teams, without the teams’ direct involvement. It is preparatory work that can be performed directly by the *Product Owner* without interfering with the teams’ work.

(We will see shortly that there is also a *Team-Side* to the *Full-Kitting* activity).

In the simulation, the *Product Owners* will also act as *Flow Managers*. They can improve the *Flow* of work by ensuring that they have properly prepared all work in such a way that teams can start working as soon as they are ready.

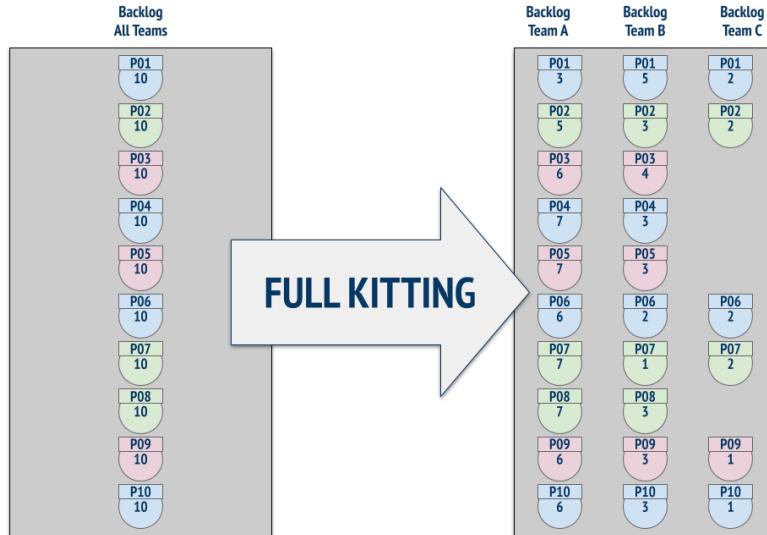
In the simulation, instead of handing out coins one by one to the teams, the *Product Owner* will prepare specific work-packages made up of coins and then hand over the coins in the work package all at once to any front line Team.

It is as if they were “preloading” the *Team Backlog*. If we just look at the first *Product Owner*’s action, it could be represented visually as follows:



More generally, if all projects were “Full-Kitted” this way, the backlog table would end up showing three distinct backlogs - one backlog for each of the front line teams - as shown in the illustration below.

The *Full-Kitting* activity at this level is transforming the perspective from the portfolio level (“*All Teams*”), to the team level (with individual team backlogs), effectively preparing the incoming work for the individual teams.



Here we see how the ten coins in Project P01 become three coins for Team A, five coins for Team B, and two coins for Team C. Then the ten coins in Project P02 become five coins for Team A, three coins for Team B, and two coins for Team C. And so on for all ten projects in the initial backlog.

Note: We are looking only at the coin distributions to the *front line* teams - so Team D is not included in setting up these simple *Full Kits* for the simulation. However, in a real scenario, a *Full Kit* would include *anything* that is known or knowable to be needed by *any* team in the *Work Flow*, even those beyond the *front line* teams. As always, we consider the simulation as a simplified *Mental Model* of reality - but not all aspects are simulated or needed to appreciate the value of the *Mental Model*.

Incidentally, we can already see that *Team A* is the most loaded and *Team C* is the least loaded. In fact, some projects do not utilize *Team C* at all. Now, after this *Full-Kitting* step, it starts to become self-evident.

At this point of the simulation, the teams would perform a run without *Full-Kitting*. Then a second round with *Full-Kitting*; and compare project durations and delivery performances. Typically, the projects executed with the *Full-Kitting* activity will be delivered faster.

Simulating Missing Information

Naturally in this first simulation of *Full-Kitting* where we did the preliminary apportionment of work to the teams, we would consider only the front-line teams, and not *Team D*, for the simple fact that the hand out of work goes from the *Product Owners* to the three front line teams - which does not involve *Team D*.

However, there are cases where the *Full-Kitting* effort must take into account whatever preparatory activity might influence the performance of *any* team - even those that come after the front line teams.

One such instance is in preventing the disrupting effects of *missing information*, which might be discovered only while work is in process. In the simulation, we will replicate the disruption of missing information, and then see how much impact *Full-Kitting* can have on this.

We are trying to simulate the case when - while active development is in progress - it is discovered that some critical piece of information is missing. It could be that the *Full-Kitting* was insufficient, or that some new facts became known. This interrupts *Flow*, as information has to be requested upstream. And because the actual information is not available, we have to resort to a *Flowback*.

As we learned, the way to handle *Flowbacks* is not materially to do the flow back, but to let the item wait in place and get the upstream resources to move downstream and resolve the issue. This works pretty well when those upstream individuals are within the same team.

If - as is the case here - the *Flowback* crosses some organizational boundary (i.e. its destination is under a different reporting line, or outside the organization altogether), then by letting the item be in place, it will become painfully evident how work *ageing* will deteriorate quickly.

Note however, that in the simulation we will actually do a real *Flowback* at first to see its effect, and then rerun the simulation with the item staying in place.

When such requests for help and intervention cross organizational or temporal boundaries, it is not uncommon for the “hiccup” to take weeks to complete the cycle and get resolved. A simple piece of missing information can induce an avalanche of negative effects on *Wait Time*, *Flow Efficiency*, *Work in Process* and ultimately *Throughput* and loss of profitability.

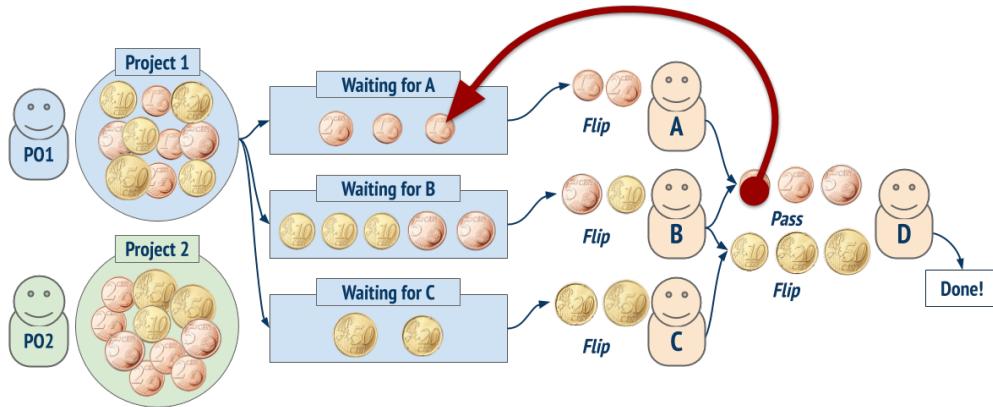
Round 1 - Heads or Tails as Bearer of Information

We want to simulate this painful situation, and then see how we can mitigate or avoid it with more thorough *Full-Kitting*. We start by making sure that coins are placed haphazardly, with the face of the coins to be randomly heads or tails. The *Product Owners* will pick up the coins from all their projects, drop them back into the respective project containers, letting them fall randomly on whatever side gravity decides.

We let participants run the projects and log the metrics. This will serve as a baseline.

Then we run the project again, with the same procedure. So all coins have a random head or tail facing up. But this time we introduce a new rule. Team D, when performing the quality assurance step, must make sure the coins match a special requirements provided by the customers. One of these requirements is that in the final configuration acceptable for delivery, all coins must be heads up - to symbolize that certain information was taken into account; and that is the information that must be visible on all coins for the final delivery.

Flow Back due to Missing Information



Team D can only work according to rules of its own *Work Process* by flipping all gold coins, and *not* flipping the bronze coins. It cannot “correct” the situation by flipping bronze coins or avoiding to flip gold coins. If there is a mismatch, the coin is sent back so that it will bear the right information when it returns back to the final quality inspection step.

Notice that Team D will check if all coins are heads up only *after* it has executed its part of the *Work Process* by flipping the gold coins. If any *single* coin is not with the correct face up, then *that single coin* will be sent back - individually - for corrective action. It will be sent back to the *Product Owner* of the project (positioned in front of the backlog table). There the *Product Owner* will reposition the coin so that it bears the correct information that can then be validated at the end of Team D’s quality control step.

Note: If this seems like a *Flowback*, it is. Subsequent runs could simulate the more appropriate way of handling such issues, with swarming onto the point of discovery of the missing or defective information.

Any coin that is being repositioned by the *Product Owner* (the face is carrying the correct information that was missing) will have to be reprocessed by the front end teams as well and go through the whole *Work Process* before they reach Team D for the final steps, including the quality inspection. Remember that coins have to be correctly associated with their project as this is the part of the delivery requirement.

In other words, the *Product Owner* can not just fix a coin and give it back to Team D. The coin will have to be given to the appropriate front end team for reprocessing and then handed over to Team D for final handling and quality control. This simulates the fact that the correct information might affect all steps in the overall *Work Flow*.

Notice the subtlety here. The *Product Owners* need to think about how to place back the coins. The

bronze coins will undergo *one* flip in the *Work Process*, performed by the corresponding front end team, and no flip by Team D. On the other hand, the gold coins will undergo *two* flips before they reach the final inspection step, a first flip by the corresponding front end team, and then a second flip by Team D.

The Product Owner has to be careful to place all bronze coins tails up, while all the gold coins must be heads up. This simulates the fact that adding information into a *MOVE* item has to be done with care.

This will also create a further housekeeping challenge. When the single coin flows back and has to be reprocessed, the front end teams will most likely be busy working with the coins of several other projects. We start to have issues about interrupting, rescheduling and resuming work, and about priorities (which are resolved by the initial ranking of the projects- a higher ranking project always has precedence over a lower ranking one.)

Even Team D will start experiencing higher levels of multitasking. As it sends back the coin for one project, it will start work on another project. The second project too will have coins with the wrong side up, and they too will be sent back. Then Team D starts working on the third project, and so on. At the end, Team D will be overwhelmed with multitasking, and just keeping track of returning *Flowbacks* needing to be reattached to projects that are still waiting. Concurrently, they must respect the relative business value priority. It is going to be a nightmare for them!

Flowbacks induce multitasking even when things appear to be well organized. And with that we will measure a dramatic deterioration of duration and due time delivery.

Round 2 - Avoiding Flowbacks

The participants will then engage in a second run with the same setup, but this time Team D will not perform a *Flowback*. Instead it will call the upstream specialists to their desk. First Team D will call the *Product Owner* to fix the information (and turn the offending coin so that it has the correct side up). Then Team D will call the front end team in charge of that kind of coin to perform its processing step. Finally Team D will flip that coin (or not flip it depending on whether it is gold or bronze), and perform the inspection step. All of this will be repeated for every single coin that has the wrong side up, until they all have the correct face up.

This second round might not run that much faster, but the housekeeping burden should decrease. The coordination and synchronization costs are lower.

It is interesting to observe that with this way of working, we are actually interrupting the work of Team A - which we know is the *Constraint*; and we know that the *Constraint* should never be interrupted.

However, if we now ask the canonical question: “*Where is the Constraint?*” - What would the answer be? We let the participants ponder and maybe schematize the queues of work in front of each Team. Clearly, when a coin has the wrong side up, it becomes an action item for Team D. Assuming that coins have a 50% odd of being with the right side up, we realize that Team D now has to reprocess half of all coins by sending them back, in addition to flipping its own coins as per the original *Work Process*.

After doing some math we would see that with this particular shape and form of demand, Team A is effectively still the *Constraint*. Therefore, we should try to prevent this from happening because the overall impact will be an even more reduced *Throughput* because we will have to interrupt Team A to do corrective work.

There is an interesting effect to observe. The induced rework has increased the *Work Load* of all teams, but in relative terms, Team D's *Work Load* has increased dramatically. While for the front line team the load factor increases by approximately 50% (remember the odds of a coin being on the wrong side), Team D's load factor increases by almost three and a half times.

That is how detrimental missing information can be.

We can also see that if the situation was more dramatic, missing information could even move the *Constraint* in the *Work Flow* to Team D. So we have a very good reason to prevent this from happening.

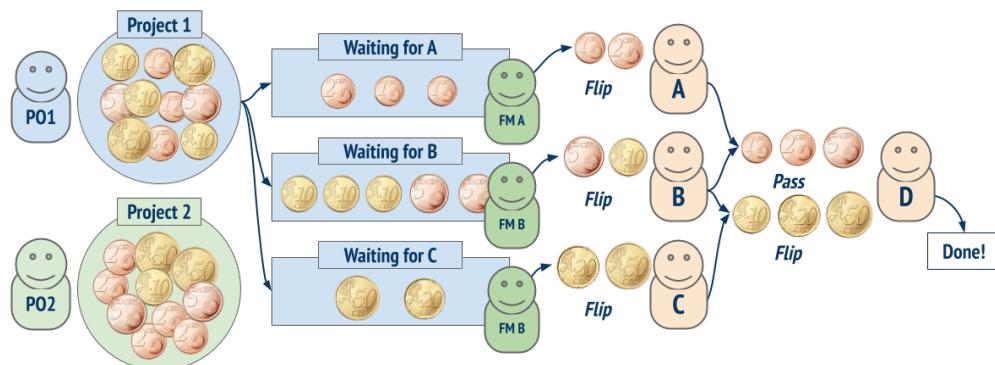
Round 3 - Simulating *Full-Kitting* on the Team Side

In order to avoid missing information, the team will accept work only if the provided information (requirements, specifications, etc.) meet minimal acceptable standards. To this end, we will consider the activity of *Full-Kitting* on the team side.

There will be someone who is familiar with the *Work Process* of the team and capable of providing feedback on whether a project (a *MOVE*) is complete and ready for being accepted for work by the team. Team side *Full-Kitting* ensures quality input to avoid unnecessary defects and rework.

In a way, a *Flow Manager* on the team side ensures that the team has one “public” interface, a designated contact person. This role can also act as a watchdog, ensuring that nobody skips the line (as established by the prioritization ranking) so that the team is not artificially overloaded. The team side *Flow Manager* will inspect all inbound work to ensure that it has all what is needed by the team for completion.

We can visualize the team side *Flow Manager* - in green - positioned as shown in the following illustration:



With this new role in place, the simulation is run again. The team side *Flow Manager* will inspect the incoming coins. If they find any coin showing the wrong side, they will ask the *Product Owner* to fix it immediately. (The entire project will be sent back to the *Product Owner's* backlog for fixing.)

By doing so, wrong sided coins will not even enter the system, and the quality control performed by Team D will truly catch only the exceptions that might have skipped through the checks.

By running the simulation with this configuration, *Flow Time* will improve dramatically. Even with such improved times, all coins will bear the right kind of information. There will be very few instances of missing information (i.e. coins with the wrong side up).

Takeaways

The *Full-Kitting* activity is typical of the manufacturing world, and is easily dismissed as inappropriate in knowledge-work. However, considering that organizations have plenty of *Excess Capacity*, it is clear that such capacity can be proficiently employed to improve how the incoming *Work Load* will flow through the system. *Full-Kitting* has the purpose of creating the conditions for *One-Piece-Flow*, avoid interruptions, blockages, *Flowbacks*, and generally reduce the cost and time of coordination and synchronization.

In the next chapters we will see that *Full-Kitting* can become a full fledged executive activity that truly sets the foundations of high performance *Operational Flow*.

Jerzy Stawicki says...

This book strives to combine Kanban and the Theory of Constraints in one seamless approach. Can it be done? Yes! Steve and Daniel have done it. They focus on the best of these two worlds and link them together in a very practical way, especially for multi-project environments.

The result? A fascinating book, clearly written, which you read almost as Raymond Chandler's hard-boiled detective novel.

Want to know what Jeep-Jungle-Journey metaphor is about? Want to have tons of inspiring thoughts and ideas to implement in your company to get results and achieve business agility?

Just read and study "Tame your Work Flow" by Steve and Daniel. And then go and implement their ideas in practice.

Jerzy Stawicki, PhD Organization and Management, Management 3.0, Kanban/Agile, OKR, PM consultant & trainer

PART 6—Achieving High Performance Execution and Governance

In PEST environments, we are now capable to identify the *Constraint* in the *Work Flow* by looking at the longest queues in front of the teams. In the *Constraint* team, we can find the *Constraint* in the *Work Process* simply by looking at the longest average in-state *Flow Time*.

We also know how to prioritize and select projects.

Yet when it comes to executing projects, we must contend that *Mr. Murphy's* visits will steer us away from our path. So, like the Boy Scouts, we need to be prepared - prepared even for surprises.

In this part we will learn about how to “*prepare to be prepared*” with a structured way to perform *Full-Kitting*. Then we will learn how to organize work in *MOVEs* (packages of work that we can manage according to the principles of the *TameFlow Approach*.)

Finally, we will master the science required to identify and act on the *Constraint* in the *Work Execution*, even when there might be hundreds of concurrent projects, by looking at buffer management at scale.

We will see how all these elements come together and allow us to address the bigger picture - the one that top-level managers care about - and how we can support the decision-making priorities they face. We know how *Management Attention* can be a critical *Constraint* in organizations. Managers are overloaded with information; their attention is stretched in all directions; they are short of time. We will provide leading indicators as effective *governance* tools. Management will now be in the unique position to act preemptively with a razor sharp focus to address all the *Constraints* in the system.

Vasco Duarte Says...

I often get asked by clients: "How can I make my engineering faster? We're too slow at responding to competitors and introducing good ideas to the market!"

The answer is simple: "You have no flow in your organization!" What does that mean in practice?

This book goes into the nitty gritty of what Flow is and why it provides (when improved) massive Business Value! Simple techniques and practices like postponing commitment and limiting work in progress become obviously good ideas when you read this book!

As a bonus you get loads of stories about how certain patterns contribute to improved flow! A must-read!

Vasco Duarte, #NoEstimates author, Scrum Master Toolbox Podcast host

18—Full-Kitting as Ongoing Executive Activity

We discovered the value of *Full-Kitting* as a means to ensure that work flows in an uninterrupted manner through the system. The activity happens “upstream” - before work actually enters the design, implementation and delivery phases. *Full-Kitting* appears to be very similar to *Planning*. In practice, *Full-Kitting* replaces conventional detailed planning entirely, but it is not the same thing.

Full-Kitting should instead be considered as an *executive* and not a *planning* activity - and even more so in *PEST* environments. When there are so many moving parts (**P**rojects, **E**vents, **S**takeholders, and **T**eams) and so much *VUCA*, there simply is no time to draft, approve and perform conventional project plans. Therefore in the *TameFlow Approach*, the *Full-Kitting* activity is considered part and parcel of *Work Execution*.

Given how we have been building our *Mental Model* on how to operate in a *PEST* environment, the scope of action of the *Full-Kitting* activity covers a single *MOVE* at a time. One might wonder, what essential elements must be part of a *Full-Kit* before actually committing to a *MOVE*? Remember the *Committed* column in the *Portfolio* board we saw in *Chapter 12 - Drum-Buffer-Rope Scheduling*? Well, we must clearly establish what must be known about any *MOVE* before it can be accepted in the *Committed* column of the *Portfolio Kanban Board*.

Failure Demand

No book on knowledge-work would be complete without addressing *Failure Demand*. Professor John Seddon first made the distinction between *Value Demand* and *Failure Demand*.



Value Demand: demand (i.e. *Work Load*) where customers pull value from the organization.

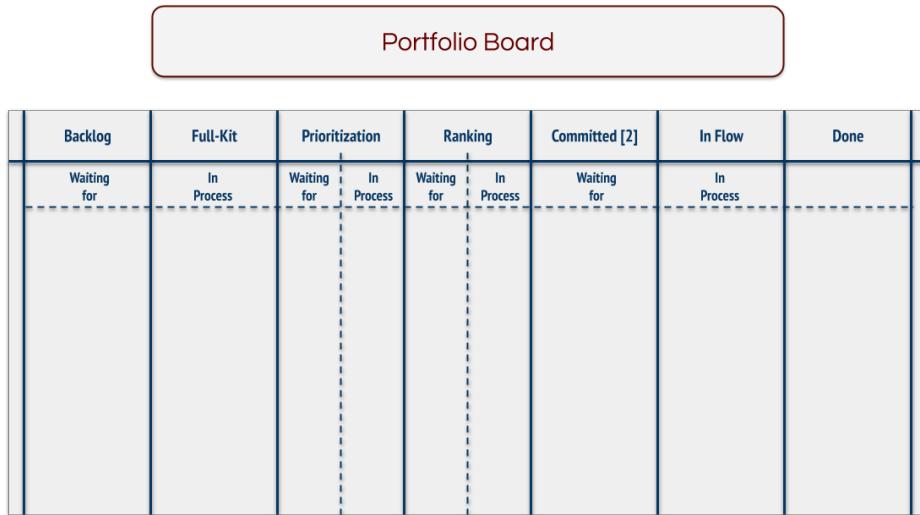


Failure Demand: demand (i.e. *Work Load*) caused by a failure to do something or do something right for the customer.

This book deals with new *Mental Models* that are aimed specifically at curbing *Failure Demand* and nowhere else is John Seddon’s observation more appropriate than in this chapter. *Full-Kitting* is not only something that must be done, it must be done right. *Failure Demand* has a big impact on costs in terms of rework due to poor business requirements. Upstream is the place where due diligence and care is required to minimize the most difficult component of *Failure Demand*: business requirements with a concern for *Flow*. Evidently, other forms of *Failure Demand* exists such as rework, cancelled work, poor usability, invisible work and all the ailments caused by excessive WIP etc. *Full-Kitting* is the preventive cure to curb *Failure Demand*.

The ***Full-Kitting Work Flow***

The *Full-Kitting* activity has its own upstream *Work Flow*, which precedes the ordinary *Work Flow* (where “stuff gets done!”). We capture this *Work Flow* by representing it explicitly on the *Portfolio Kanban Board*, as follows:



Let us see what steps are involved and what happens.

The ***Backlog Column***

The *Backlog* is simply where requests first land. Ideally it is a queue that precedes any activity. *MOVEs* are deposited at the bottom, and picked up at the top. This is a state that incurs *Queue Time*. It is waiting time - though it is outside of the actual *Flow Time* - so it does not impact *Flow Efficiency*.

Note that when the *client-side Flow Managers* add a new *MOVE* to the *Backlog* column, the *MOVE* needs to indicate what the *customer's requested due date* is.

Whenever the *Flow Managers* meet to decide which *MOVEs* should be transferred into the next phase, they will typically do so on the basis of the desired due date, giving precedence to those items that are due sooner.

The due date can be determined in many ways, but at this phase it should reflect what is desired by the customers. Once the actual prioritization activity happens, dates can be overridden because of other priorities, policies, *Cost of Delay*, or other criteria - and in particular the *Throughput Rate* as we learned in *Chapter 13 - Portfolio Prioritization and Selection in PEST Environments*.

The *Full-Kit* Column

The *Full-Kit* column is where the *Client-Side Flow Managers* interact with the actual teams, (possibly through the *Team-Side Flow Managers*) to break down the selected *MOVEs* into actionable pieces, like *User Stories*, *Features*, or anything else that the teams recognize as reasonable and feasible. A single *MOVE* might need the contribution of more than one team (after all, we are in a *PEST* environment), so there might be a multitude of *Team-Side Flow Managers* contributing their insights.

Note: If the *MOVE* is so elaborate that the teams' opinions cannot be gathered in a quick and simple way (i.e. they might require further work, research or prototyping), then that very exploratory activity should be represented by a *MOVE* of its own (a "spike" in eXtreme Programming terms), with higher priority.

This is also where any other piece of information, equipment and third party supplies need to be coordinated and confirmed to become available the moment they are needed during the teams' execution of the *MOVEs*.

Once all elements are available, the *Flow Manager* will use probabilistic forecasting to establish the *duration* of the *MOVE*. Notice that this effectively requires *two* forecasts:

- a forecast of the *MOVE* duration, so that reliable due dates can be promised to the customers; and
- A forecast of the *MOVE*'s engagement on the *current constraint in the Work Flow*, so that prioritization can be calculated.

Once all the forecasts are expressed, the *MOVE* will proceed to the *Waiting for Prioritization* column.

Note: If our organization is not equipped to perform probabilistic forecasting, then lightweight estimation techniques should be employed (like *T-Shirt Sizing*). Any other more elaborate expert opinion-based estimation techniques would require too much time to be viable. This is why it is preferable to have automated probabilistic forecasting for this phase, leveraging existing *Flow Metrics*.

The *Prioritization* Column

On a regular basis - typically once a week - all the *Client-Side Flow Managers* (there are more than one, since we are dealing with multiple *MOVEs* and multiple stakeholders), will meet to select the *MOVEs* that are ready for prioritization and actually put them in priority order.

The *MOVEs* will already show a calculated priority, derived from their engagement of the constraint and their business value, for instance using the *Throughput Rate*. Note well: this is calculated automatically because there are so many *MOVEs* that figuring out the sequence manually would be an exercise in futility.

Note that the *calculated priority is not the definitive one*. It is only a suggested one. There will always be human factors, compromises, and exceptions to deal with. Once the team of the *Client-Side Flow Managers* has selected the *MOVEs* it wants (on the basis of both the calculated priority *and* their knowledge of any other stakeholder or market conditions), there are two possible scenarios:

- The calculated ranking is fully acceptable. It then passes directly into the *Committed* column and is placed in the queue in the position determined by the calculated prioritization.
- The calculated ranking is not adequate, for other non-technical reasons.

The second scenario could be caused by some immovable deadlines (and *Cost of Delay* considerations could be the tie-breaker). It could also be due to special customer-relationship issues (like a smaller project for a major customer that is too important to be ignored). All such non-technical problems are known to the *Client-Side Flow Managers*. They have to represent the issues in the meeting. The *Client-Side Flow Managers* will negotiate the resolution of priority conflicts between themselves.

The *Ranking* Column

If a resolution cannot be found during the session, then the *MOVEs* under consideration are submitted directly to executive management with a request for ranking. It is necessary that the items that did not receive a clear priority placement be passed on to the attention of executive management.

This typically happens the day after the prioritization meeting. At times though, if the situation is complex, the request for ranking might need to be further elaborated with factual information for executive management, and could be postponed to the same meeting of the coming week. In any case, the idea is that executive management will receive a *request for ranking* with no material delay and, especially, containing all factual information - in order to swiftly make a decision. Once a *Rank* has been assigned by executive management, the *MOVE* is placed accordingly in the *Committed* column.

This mode of operation *requires the active involvement of executive management in shaping demand*. This is unlike most other managerial approaches where executive management is typically concerned with forward looking strategic issues and after-the-fact evaluations, but not at all involved in sustaining the organization's *Flow*. With the *TameFlow Approach*, they become an active part. Management's active engagement is one of the distinguishing features of the *TameFlow Approach* that sets it apart from other Agile activities - like the *Backlog Refinement* or *Sprint Planning of Scrum*.

Interestingly, *Throughput Accounting* can again come to the rescue in this case and initiate structure. The *Throughput Rate* metric plays a crucial role. It allows us to compare "apples to bananas" and to be able to get better at forecasting, *MOVE* construction, gauge projects across divisions, products and regions among other things. It will also reveal our true capabilities and *Capacities* so that we can stop using ROI's pink colored glasses when deciding what project to initiate. Sometimes, in order to change *Mental Models*, one needs to be supported by new and more realistic metrics.

Note: It is necessary that *Executive Management* understands its role in light of the *Mental Models* explained in this book. Once management realizes that with little extra effort, the speed of its operational decision-making can have a significant impact on the bottom line, its participation will be easy to secure. Remember the example of *Chapter 13 - Prioritization and Selection in PEST Environments*, where changing ranking criteria from ROI to TR had an impact of over 30% more *Throughput*. With that

kind of argument, top managers will find the motivation to play their part.

The *Committed* Column

When *MOVEs* find their queue placement after the *Prioritization* phase or the conflict resolution and tie-breaking *Ranking* phase, they are moved over to the *Committed* column. Once a *MOVE* has been placed in the *Committed* column, its position will not be changed.

Committed is committed, no matter what.

At this point most will argue that there will always be *urgent* items that need to be ranked higher. Well, let's remember how disruptive that was in the simulation, when the *Product Owners* were able to skip the line? The rationale that needs to be fully subscribed to - and actually strictly imposed - by *Executive Management* is that the time lost due to the disruption of skipping the lines is much more than what could possibly be gained even considering special treatment of urgent items.

One corollary here is that we want to keep the number of items in the *Committed* column as small as possible, because that leaves more options to pick the next *Committed* item. If the *Committed* column is small enough, usually placing an urgent item (which nonetheless must go through all the prioritization and ranking processes described above!) at the bottom of the *Committed* column will produce much less damage and overall disruption than the line-skipping malpractice. And because execution is optimized to avoid *Wait Time*, the total time for the urgent *Work Item* to wait in line in the *Committed* column and to flow through the *Work Process* is usually less than the time lost due to coordination and synchronization of all the teams involved. What is often not understood, is that when an urgent item is made to skip line, *not all* of the teams can become available at the same time (bottlenecks due to non-instant availability). Consequently, even the urgent *Work Item* will incur *Wait Time* inside the process whenever it breaks it!

Note: As we saw in *Chapter 12 - Drum-Buffer-Rope Scheduling*, one way to determine the appropriate size of the *Committed* column is to consider it as a proxy of the *Buffer* column in the *DBR Scheduling* around the *Constraint* in the *Work Flow*. This is a policy that we can adhere to even if the *Buffer* might appear downstream in the *Work Flow*, and not in front of it. So while we want to keep the *Committed* column as small as possible, if it is also acting as the *Buffer* of the current *Constraint* in the *Work Flow*, then we also have a *minimum* number of *MOVEs* that it should contain. The sizing of the *Committed* column should thus not be less than the size of the *Buffer*.

The *In Flow* Column

Finally, when the *Constraint in the Work Flow* emits a *Replenishment Signal*, the topmost *MOVE* in the *Committed* column will be released into the *Work Flow*, and placed in the *In Flow* column. The individual front-line teams will pull the *MOVE* into their own team-level *TameFlow Boards*, and subject their part of the *Work Load* to their team's *Work Process*.

Operational Full-Kitting

When there are many *Stakeholders* represented by many *Flow Managers*, all their projects and initiatives can be managed with their own *TameFlow Boards*. The *Portfolio Board* we have encountered in this chapter must be considered as the *master board*. When the *Client-Side Flow Managers* collect the requirements and desired data, then *all* those *MOVEs* should go onto the *Portfolio Board*. It is important to realize that if we want to be able to manage the *Work Flow*, then *all work must be made visible, at all levels and at all times*.

The *Portfolio Board* becomes the single version of the truth - a tool of absolute and enforced transparency. C-level executives will institute a company wide policy that if some work does not originate from the *Portfolio Board*, then teams will simply be prohibited from doing it. It is that simple. If we want to manage all work, then all work must be visible. Unless we do this, we will not be able to scale organization wide.

It is through this absolute transparency that we will detect issues and blockages, and enable quick interventions with unprecedeted focus and foster deeper collaboration and instill the patterns of *Unity of Purpose* and *Community of Trust*. The company wide policy of absolute transparency can - in turn - make the individual customers' or *Flow Managers*' boards simpler. Rather than having their own boards, they can just use the *Portfolio Board* and filter it to show only the items that they are responsible for. From the clients' perspective, the filtered *Backlog* column is where they write their wish list. The *Committed* column is where they will see which of their *MOVEs* have been accepted by the organization, with the due date forecasts.

The *In-Flow* column can decorate all *MOVEs* with the *Buffer Status*. (see *Chapter 16 - Introduction to Execution Management Signals* for more about the *Buffer Status*.) The *Buffer Status* could be rendered graphically with a red-yellow-green traffic light, which could also turn to black to show overdue items. This, combined with the focused governance practices described in *Chapter 20 - Operational Governance in PEST Environments*, will give the Client-Side *Flow Managers* all the information they need to report back to the Stakeholders about how their parts of the overall portfolio are progressing.

Takeaways

The activity of *Full-Kitting* is fundamental when dealing with PEST environments. It is the moment when new requests are examined together with the teams to build a collective understanding of what work is about to come. It is when the teams get prepared for the work so that it can be handled without interrupting its *Flow*. Even more importantly it is when the prioritization and selection decisions are made. First automatically based on a tentative *Throughput Rate* scoring; then collegially via discussion of exceptions among the *Flow Managers* who have broader knowledge of other relevant factors. If the *Flow Managers* do not reach an agreement, resolution by top management is the final step. Involvement of top management is swift and focused; yet it is what distinguishes the *TameFlow Approach* from most other methods or processes where top management is barely involved at all - and if it is, it is only after the fact at performance evaluation time.

Finally the *Full-Kitting* activity, while scheduled on a regular weekly cadence, is also governed by *DBR Scheduling* across the entire portfolio to decide how much new work can be released into the system. It is when making such decisions that contingent *Capacity* issues are taken into consideration (as will be described in later chapters).

Daniel Hernández says...

Having previously read the excellent “Hyper-Productive Knowledge Work Performance” from the same author my expectations on this book were high, and it didn’t disappoint. As far as I’m aware, “Tame your Work Flow” represents the first and only holistic attempt to gear knowledge-work organizations to high performance.

By holistic I mean using Systems Thinking to look at the organization from four different prisms (or flows): operational, financial, informational and psychological.

To do so, authors Steve Tendon and Daniel Doiron have masterfully decoded Eli Goldratt’s Theory of Constraints body of knowledge to then apply it in complex settings involving multiple teams and value streams.

The success of this book lies in making the complex look simple, thanks to the use of Mental Models that explain the theoretical concepts behind the TameFlow Approach.

But don’t be mistaken with the word theory, this book is eminently a practical one. Packing numerous examples, training games and patterns to help you get started with TameFlow in your organization.

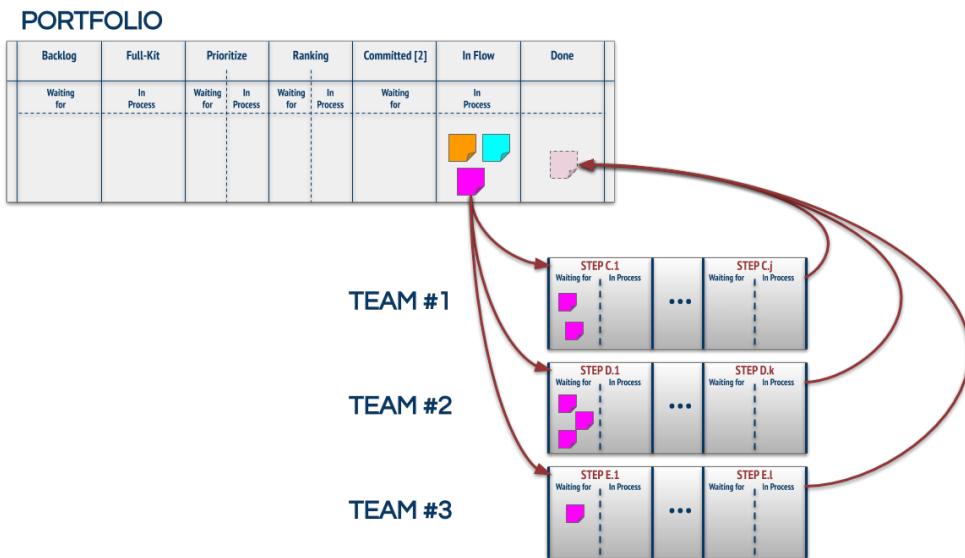
I thoroughly enjoyed reading and learning with “Tame your Work Flow” from beginning to end, and I believe you will too!

Daniel Hernández, Scrum Master.

19—Execution Management in PEST Environments

In the preceding chapter we saw how the *Full-Kitting* process produced a prioritized list of *MOVEs* in the *Portfolio* board's *Committed* column. Once a *MOVE* item is brought from the *Committed* column to the *In Flow* column on the *Portfolio* board, it will expand into distinct *MOVE* items - one for every team that needs to work on it. Then once a team pulls its version of the *MOVE* item, the item is expanded into the actionable items (like *Use Cases*, *User Stories*, *Features*, etc.) that the team will be developing.

Schematically it looks like this:



Here we see that the *Portfolio Manager* will be concerned about the progress of three *MOVEs*, distinguished by color for illustration purposes - an orange, a cyan and a pink *MOVE*. We also see how the pink *MOVE* is broken down further and assigned to three teams, operating either a *Flow Efficiency*, a *DBR* or a *Throughput Management Board*.

A pattern begins to emerge. The *Portfolio Manager* is responsible for one portfolio, which can contain many *MOVEs*. Similarly, the *Flow Manager* in charge of the pink project is responsible for how it is moving across many teams. The pattern is that someone is responsible for *many* moving parts, even if it happens at different levels in the organization.

This is where we need a smart way to monitor the overall status of items moving through the system. Most importantly, we want an overall picture of how many items are moving along in order to reach a common outcome. And we need to be able to react quickly when something is “going sour.”

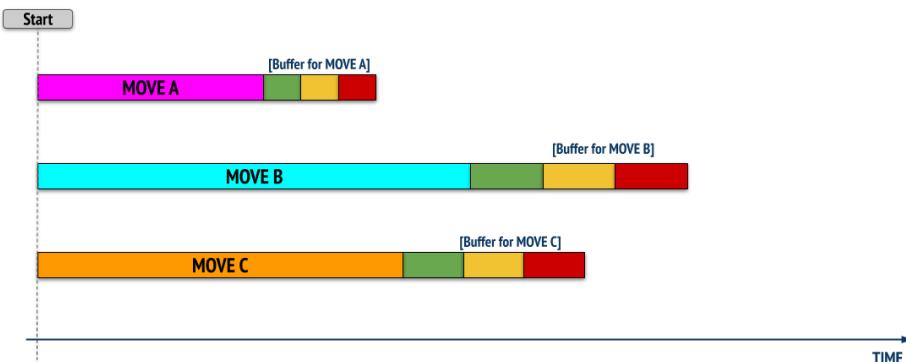
In the remainder of this chapter we will focus on the *Portfolio level perspective*, and we will consider the three *MOVEs* discussed above.

Many Moving *MOVEs*

We will assume that every *MOVE* has its own *Buffer* established as we saw in *Chapter 16 - Introduction to Execution Management Signals*. We will also assume that the three *MOVEs* are independent, in the sense that they do not need any material integration.

If the projects start at the same time, they will all have different timelines, with their respective buffers placed wherever their timeline ends. It might look like this:

MOVEs might appear independent...



Virtual Integration Points

Why assume that the *MOVEs* are independent from one another? In reality they will often have to share resources along the timeline. In particular, we can consider that there will always be events and deadlines against which the *MOVEs* need to synchronize - even if the *MOVEs* happen to be truly independent.

There are various possible deadlines:

- commercial (such as trade shows)
- internal business reporting periods (like quarters and end of years)
- regulatory deadlines (like fiscal years and tax or VAT declarations) etc.

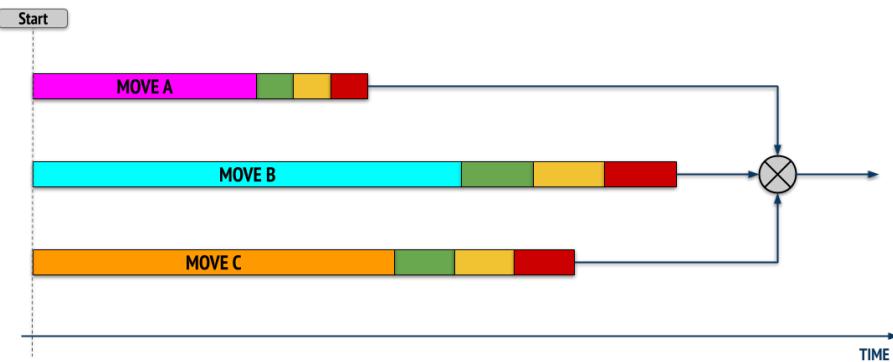
In all cases, since the system is loaded with many projects requested by so many different stakeholders, the interlocking effect of all of them combined is that such **Events** (the “**E**” in the PEST environment) become de-facto *synchronisation points*. All, or a great number of projects need to be ready by such deadlines, hence the strategic value of coordinating all activities so that they finish on time, as if they were not independent.

For example, there can be a major industry sector exhibition and we happen to be providing components to all the major players in the industry sector. They all want to show off their new products at that event. And even if all of them are signing different contracts with us, and all of them have independent requirements, needs and deliverables, that deadline becomes a moment in time when most of our running projects must be delivered simultaneously.

The larger the business and its *PEST* environment, the higher the likelihood that there will be such *virtual integration points*. When considering how to bring some sanity in managing a *PEST* environment, we must assume that such *virtual integration points* always exist. Even if they really do not exist, we should define them nonetheless, because we can focus around such points, making the management process easier.

Therefore, if we simply postulate that there always is an integration point (either real, virtual or purposely created) our three *MOVEs* timelines will all converge on the integration point. The previous diagram can now be updated as follows:

... but where there is PEST, there is always an integration point, even virtual!



Planning with Respect to the Constraint

If we need to plan and coordinate all activities with respect to the *virtual integration point*, we need to find the constraint among all moving parts. We already learned how to do this in *Chapter 11 - Finding*

the Constraint in PEST Environments. In the simulation of that chapter we looked at the *Work Load* of each team which was represented by the number of coins to be flipped.

In a real world situation, there are no coins to flip. We look at the forecasted duration of the activities. Since we have a probabilistic stance, we do not consider the durations as fixed amounts but as ranges with probabilities - the ranges that are represented by the *Buffers*.

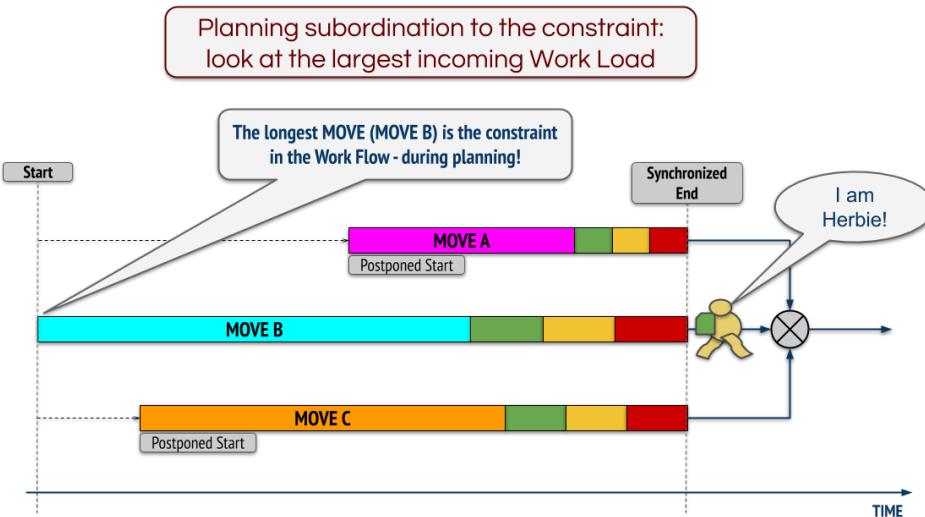
Also, we must be aware that probabilistic forecasting based on *Little's Law* will only work if the system is stable (remember the parallel lines of demand and delivery in the first chapters). This in turn implies that the teams actively avoid *Multitasking* and actively minimizes *Work in Process* (via *DBR Scheduling*).

This is an extremely valuable simplification, because then all teams can be assumed to work *sequentially*, one *MOVE* after the other.

This allows us to pile up the *Target-Scopes* of successive *MOVEs* on the vertical axis of our overall team level *Work/Time* diagrams (as we saw in *Chapter 16 - Introduction to Execution Management Signals*).

By being sure that any one team is working on only one *MOVE* at a time, and that we can forecast its duration in probabilistic terms, we can proceed to the next important step. We can identify the team that has the longest queue in front of it (like we did with the coins) as the *Constraint* in the *Work Flow*. These queues are not the line of coins, but they are the the forecasted duration (ranges) for the *MOVEs*.

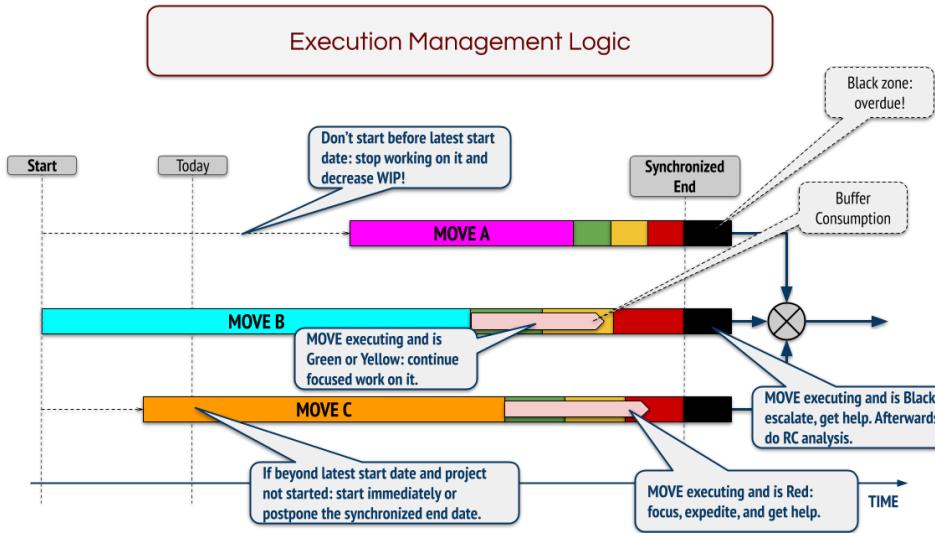
At this point we align the *End Dates* of all *MOVEs* so as to coincide with the *virtual integration point*, and visually it would look like:



Monitoring the *MOVE Buffers' Consumptions*

When work starts on the *MOVEs*, we monitor the progress and the *Buffer Consumption* for each one of them. For the sake of illustration, we can show the *Buffer Consumption* visually with a bar that penetrates

the buffer from left to right, as shown in the following figure:



In the figure we also depict the timeline after the aligned end as a black overdue zone - just to make the deadline stand out even clearer.

There are some critical considerations that we can now draw from observing decision/time-points with respect to a hypothetical “today”:

- Even if there is *Excess Capacity*, a team should not start working on a *MOVE* before the latest start date established by subtracting the forecasted duration (including buffers) from the synchronized delivery date. This should not be done because it would most likely introduce *Multitasking* in future activities; and possibly reduce *Protective Capacity* that might be needed for other recovery purposes (this wouldn't apply if the team is the *Constraint* team).
- If the execution of a *MOVE* has not begun after the latest start date, then it must begin immediately. The synchronized end date might also need to be postponed accordingly. If the end date cannot be moved, then the project will (because of its late start) trigger a *Buffer Signal*.

Whenever there is a *Buffer Signal*, then we will interpret it as follows:

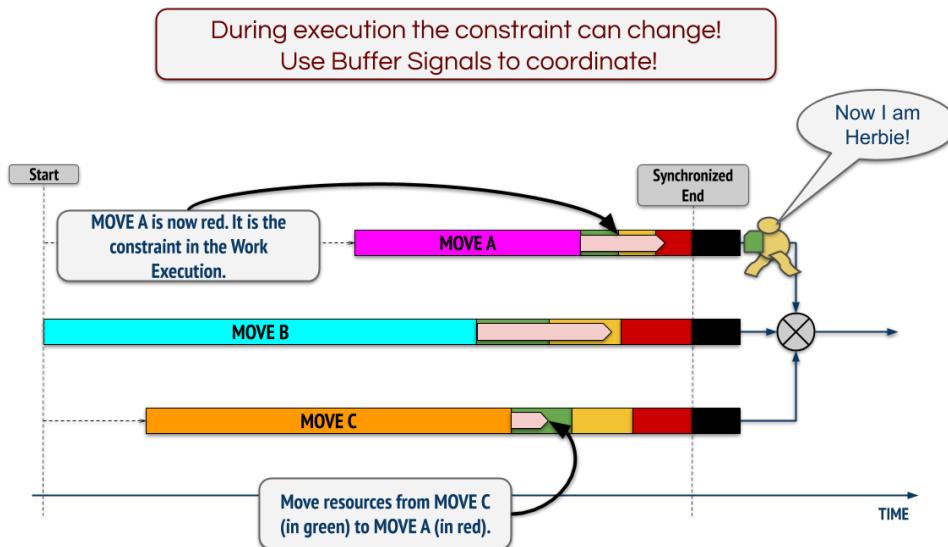
- If a *MOVE* is being executed, and the buffer penetration is green or yellow, work should progress normally.
- If a *MOVE* is being executed, and the buffer penetration is red, then work should continue with urgency and if possible, be expedited. The team should seek help from projects whose buffer penetration is preferably in a green status. A post-mortem root cause analysis should be done.
- If a *MOVE* is being executed and the buffer becomes black (exhausted), work should continue with urgency, and any issues must be escalated immediately. A post-mortem root cause analysis must be performed to identify *Common Causes* or *Special Causes*.

Particular attention is given on how to identify *Common Cause Variation* on the basis of the frequency of recurring causes.

The Constraint Caused by Execution Issues

We can now monitor the buffer consumption of all *MOVEs* being executed, and we are now in a position to detect when things go astray.

For example, examine the figure below:



We can now see the real reason why using *Buffer Consumption* for execution control is compelling. It gives us something that the portfolio management techniques of the *Kanban Method* and other Agile approaches simply do not provide.

It gives us the *leading* detection of the impact of unfavorable variation.

Notice that in the illustration, *the relative Buffer Consumption of MOVE A and MOVE C has changed* with respect to the previous illustration. This is to emphasize that in the execution world, things are dynamic and change all the time. And above all they will almost never correspond to our initial “*Plan*.”

All of a sudden, *MOVE A* appears to be challenged, notwithstanding that it had the shortest queue to begin with. It is likely that it has been impacted by some unforeseen event.

In this new situation, *MOVE A* becomes the main element that hinders the successful delivery of the whole portfolio by the scheduled end date. *MOVE A* has become the *Constraint* of the whole portfolio at this particular moment.

We qualify this as the *Constraint in the Work Execution*.



Constraint in the Work Execution: The *Work Process* that is mostly affected by unfavorable variation, to the extent that its *Buffer Consumption* is the worst in the entire *Portfolio* of currently ongoing *MOVEs*.

Unless the issues faced by *MOVE A* can be resolved immediately, the team will need top management attention and intervention.

It is through this sort of monitoring that we can detect unfavorable variability early on and intervene in time - without resorting to any dysfunctional *Column WIP Limits* that would just mask out these essential signals.

It is significant that considering the relative *Buffer Consumptions* shown in the illustration above, *MOVE B* is no longer the constraint even though it had the longest queue to begin with.

In the unfolding dynamic evolution of the execution of all projects, the *Constraint - as of the moment captured in the illustration* - has effectively become *MOVE A*, because of “things that happen.”

This is a situation where Kanban’s *Column WIP Limits* are not only masking the real issue but are actually misleading because they would just detect the longer queue (probably via exceeding the *Column WIP Limit*) of *MOVE B*, while the growing problems in *MOVE A* would remain undetected until much later.

Full-Kitting with an Eye on the State of Execution

We mentioned earlier that *Full-Kitting* needs to be considered as an executive activity. We can now see why. When *Full-Kitting* needs to determine the prioritization of incoming *Work Load*, we know we need to calculate and use the *Throughput Rate* of the *Constraint* in the *Work Flow* to establish the priorities. The *Flow Managers* will however always keep an eye on how the respective *MOVE Buffers* of the overall portfolio look like, one with respect to the other. (How we can easily do this, is the topic of the next chapter.)

This is necessary because the dynamic unfolding of events and unpredictability of *Special Cause Variation* might have such consequences that the *Constraint* in the *Work Flow* might move - and when it does, it will most likely move to the challenged *Constraint* in the *Work Execution*.

So there must always be a sense of alertness as to whether the historical *Flow Metrics* we use to compute the forecast durations are still reliable; or if the challenges we are detecting in the *Work Execution* are not temporary in nature, but might have long lasting impact on the relative *Capacities* of all teams.

Thus we have come full circle.

While the *Constraint* in the *Work Flow* is determined by the “shape and form” (statistical distribution) of the incoming *Work Load*, we must be prepared to act on the - presumably temporary - *Constraint* in the *Work Execution* the moment its presence becomes evident via the *MOVE Buffer Signals*. If the team that happens to be the *Constraint* in the *Work Execution* at the moment continues to experience problems for a longer period of time, it can be expected to become the new *Constraint* in the *Work Flow*, and thus affect the prioritization and selection of work during *Full-Kitting*.

In any case, the team that is the *Constraint* in the *Work Execution* at the moment will use its *TameFlow Throughput Management Board* to provide the drum beats of the overall *DBR Scheduling*.

Takeaways

Organizing the overall *Work Load* in terms of multiple *MOVEs* with a well defined *Target-Scope* and corresponding *MOVE Buffers* allows us to have a real time dynamic view on how *Work Execution* is

progressing.

In particular, we will be able to detect if the *Constraint* in the *Work Flow* is moving, not because of any change in shape and form of the incoming *Work Load* distribution, but because of the accumulation or extent of problems (typically *Special Cause Variation*) that dynamically impact and affect the *Capacity* of some other *Non-Constraint* resources that becomes the *Constraint* in the *Work Execution* at the moment.

With this knowledge, we are finally equipped to learn about how to exercise overall governance in *PEST* environments, as we will see in the following chapter.

Christophe Achouiantz says...

An impressive effort! Steve and Daniel rigorously apply the Theory of Constraint to the problem of Flow Efficiency for knowledge work in a VUCA context.

In search for the elusive constraint, they discover and explain in details new powerful patterns, like Full-Kitting or MOVES, that can profoundly improve any existing Kanban systems.

Moreover, what makes this book unique is that they don't stop at improving delivery systems, they also introduce Throughput Accounting and how to apply these learnings to cash flows.

Finally, the book starts with one of the most thorough and pedagogic introduction to Flow, Flow Efficiency and Theory of Constraint that I've had the pleasure to read. Well worth the read for this part alone!

Christophe Achouiantz, Lean/Agile Transformation Agent and Coach at Crisp

20—Operational Governance in PEST Environments

What we learned at the end of the previous chapter is very significant. The actual *Constraint* of a system that has to deliver a *Work Load* can also be found when examining how the *Work Execution* is performing at a given *moment* in time.

We need to reflect on this.

Constraints Everywhere!

Putting together all we have learned so far, we can see the *Constraint* from three angles:



Constraint in the Work Flow: The *Constraint* which is determined solely by the shape and form (the statistical distribution) of the incoming *Work Load*. It is literally the “*things that come towards us*.” *Special Cause Variation* is usually, but not always, manifested here. We can identify the *Constraint in the Work Flow* by looking at the heaviest queues in front of the work processes and teams.



Constraint in the Work Process: The *Constraint* is determined by “*how we do things around here*” where the emphasis is on *doing*. It is how we actually touch the work and transform it during our *Work Processes*. *Common Cause Variation* is usually, but not always, manifested here. We can identify the *Constraint in the Work Process* by looking at the longest average *In-State Flow Time*.



Constraint in the Work Execution: The *Constraint* is determined by the fact that “*things happen*.” It is the overall impact of events that occur over time - both *Special Cause Variation* as well as *Common Cause Variation* - and that affect any relevant part of the system’s capacity to deliver at a particular moment. We can identify the *Constraint in the Work Execution* by looking at the *Buffer Consumption* of the different *Work Processes* as events unfold during *Work Execution*. The one with the most critical *Buffer Consumption* is the *Constraint* in the *Work Execution*.

When we look at the big picture - the entire system - then we understand that there is only one *Constraint* in the system. The challenge is in being able to perform Step 1 of the *Five Focusing Steps* (5FS), to *identify* the *Constraint*.

The above three perspectives are just a qualification to help us rationalize as to where the *Constraint* could be, and how we can find it. In our daily life at work, we must identify and find the *Constraint*. The three perspectives allow us to apply different *Mental Models* and handling strategies.

When it comes to handling strategies, being able to distinguish between *Special Cause Variation* (SCV) and *Common Cause Variation* (CCV) is essential. Given the nature of what we are considering, CCV is more likely to affect the *Work Process*, while SCV is more likely to disturb the *Work Execution* and the *Work Flow*.

However, this cannot be generalized and we must be able to discern what kind of variation is affecting which part. That is why we need to learn how to use the *Buffer Signals*, to identify *Common Cause Variation*.

The *Kanban Method* has no provision to manage CCV, and yet, in knowledge-work CCV cannot be ignored. The *Tameflow Approach* provides the necessary tools that will make our business more profitable by knowing how to handle CCV.

No matter what kind of variation is impacting the system, we must have an optimal way to cope with it. When we are in the presence of SCV in the *Work Execution*, it can manifest simply as a momentary *Bottleneck*. We should not overreact on *Bottlenecks*, but we must be able to detect their presence and effects and be ready to intervene if necessary.

We need to develop the ability and competence to detect when a bottleneck is at risk of becoming the *Constraint*. So it is not only a matter of identifying the current *Constraint*, but also of being able to predict where the *Constraint* might move to, and especially to anticipate such a move, or even prevent it from happening altogether.

The problem is almost impossible to solve.

In large organizations affected by *PEST*, there can be tens of thousands of *Bottlenecks*, anywhere, anytime. How can we keep tabs on them, and detect if they will impact our *Capacity* to deliver or whether they might deteriorate to the point of becoming the next *Constraint*?

It is not only a matter of weeding out the small signals from the overwhelming noise. It is also a matter of knowing what the signals are to begin with. If we cannot do this, we will be misled into believing that every little blip of a *Bottleneck* is a major *Constraint*, and we will try to deploy the 5FS on them.

We have learned that if we focus efforts and resources on something that is not the *Constraint*, it will all be wasted. Applying the 5FS on *Bottlenecks* is a vain exercise in these scenarios.

Any temporary *Bottleneck* is not the *Constraint* and should not be treated as such. However, even temporary *Bottlenecks* can limit the performance enough to have a tangible impact on *Throughput*. The impact can be temporary. It might be coming from SCV that will disappear on its own. The illness of a key team member is such an example.

However, we should not ignore *Bottlenecks*. We must simply learn to not overreact and interpret their effect as a justification to start an improvement initiative. For example if the illness of a key member happens in a team that is not the *Constraint*, they might have recovered before they exhausted their *Protective Capacity* (with respect to the *Constraint*), and thus there was no need to intervene at all.

Given the ubiquity of *Bottlenecks*, overreacting will quickly lead us to improve everything, all the time. We learned in *Chapter 5 - Where to Focus Improvement Efforts* that hitting at everything that moves (whack-a-mole style) is a losing strategy.

We need to keep an eye on these *Bottlenecks* and be careful not to make the mistake of believing that they are the “*Constraint*”. If we did so, we would easily be drawn to the conclusion that we are witnessing the moving *Constraint* syndrome - and decide that *Constraints Management* is not applicable to knowledge-work foregoing all the performance gains possible (as modern Kanban mistakenly does).

We need to keep an eye on these *Bottlenecks*, because if they occur repeatedly and frequently enough, they could be due to CCV rather than SCV. Here, we are given an incredible opportunity to improve the situation with a lasting effect.

With that many potential targets of decisions and actions, we need to rewire the *Informational Flow* so that anything of relevance stands out at the right time. We need to develop new organizational nerves, neurotransmitters, and synapses to detect when and where things are happening, and to inform the right people about the right things at the right time, preferably early on.

In brief, we need some kind of *Governance* system for our *PEST* environment.

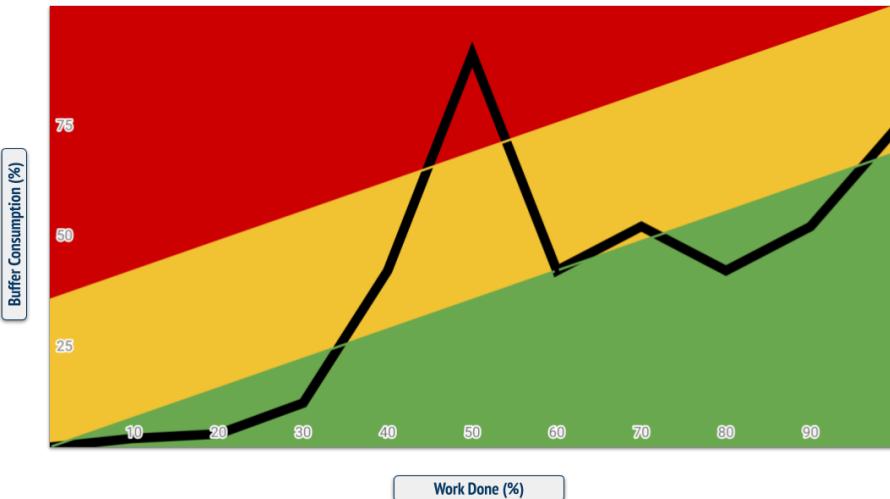
Early Detection Signals

At the end of the previous chapter - by looking at the graphical representation of the *Buffer Consumption* - we were able to conclude that the *Constraint* had moved from one MOVE to another during the *Work Execution*. This was easy to do with three projects, and with such a straightforward visual, side-by-side alignment of the project timelines, with the *Buffer Consumption* clearly shown. It is more challenging when we have to deal with tens, hundreds, or thousands of projects and initiatives at the same time.

The *Buffer Burn Rate* is an excellent indicator of how easy or difficult things are going for any particular project. Remember that the *Buffer Burn Rate* is the ratio between the *Buffer Consumption* (expressed as a percentage) and the actual amount of work performed (expressed as a percentage).

It is customary to represent the *Buffer Burn Rate* in a *Buffer Fever Chart*.

For a single MOVE, the *Buffer Fever Chart* might look like this:



This chart above represents how the *Buffer Burn Rate* has progressed over time for the full execution of a MOVE. The line extends from the far left to the far right of the chart, meaning that it went from zero to 100% of work to be done. While a MOVE is being executed, the line will extend horizontally only to the actual percentage of work done to date.

This kind of metric and chart is a great equalizer because all MOVES that are moving on the field can have a coherent, and homogenous representation of how good or bad they are faring regardless of size, budget, or schedule.

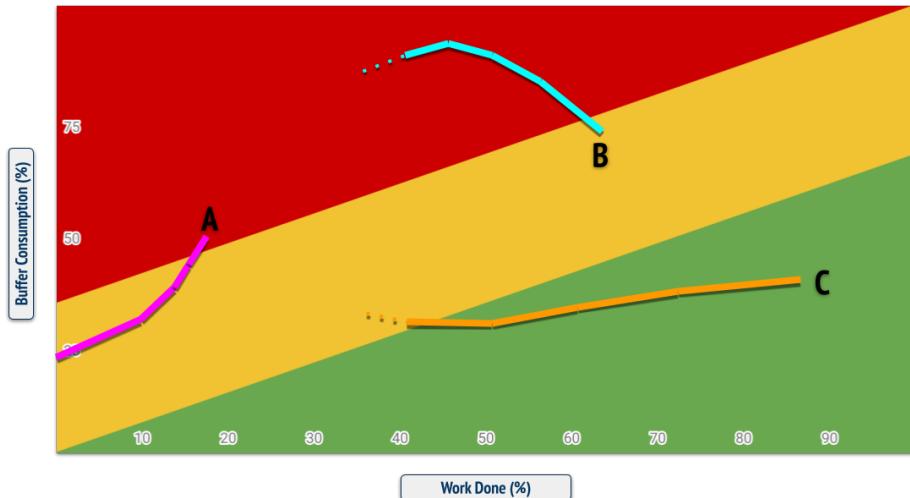
For a *Project Manager*, *Product Owner*, *Scrum Master* or *Team Member*, such a chart can be useful as a reflective tool for improvement and as a great time saver. MOVES in green should not be subjected to the same managerial attention as those in red. Don't waste people's time on trivial issues!

When we are in charge of multiple MOVES, we need a better overview. Common representations, at this point, typically include long project lists (in Excel) that are colour coded with a RAG status.

We can improve on that and instead produce *Buffer Burn Rate* charts for all projects or *MOVEs* that we're concerned with. We are recommending to focus on the last part of the respective *Buffer Burn Rate* lines of our projects, as their execution is unfolding.

We can even plot the lines of all the different projects directly on the *same chart*.

It might look like this:



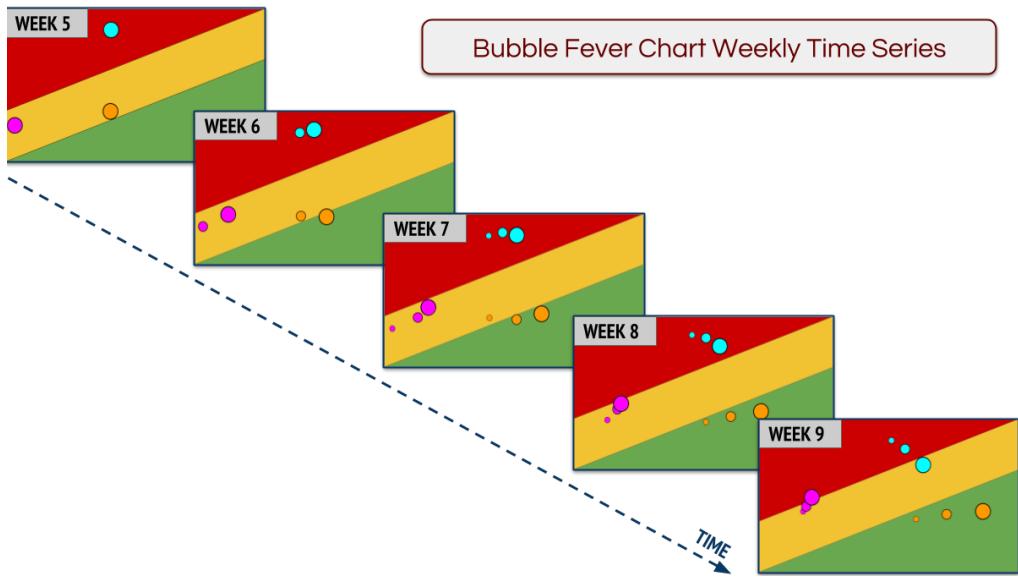
Here we are just showing the last part of each *Buffer Burn Rate* line of each *MOVE*, so we can distinguish between the three represented *MOVEs*.

Again, if we are in charge of tens, hundreds, or thousands of organization-wide initiatives, this visual representation - while still leaps ahead of any other oversight method - does not give us the full depiction of how these projects are evolving dynamically.

It is a static image. It is not a portrayal of how the *MOVEs* are, well, *moving* along.

This is where we introduce the *Bubble Fever Chart*. It is a rolling chart that is updated periodically, and which is best presented as an *animated* chart. Every frame of the animation represents a snapshot of the *Buffer Burn Rate* data points (through time) of every *MOVE* that is monitored.

For example, on a weekly basis the frames of the animation might look as follows:

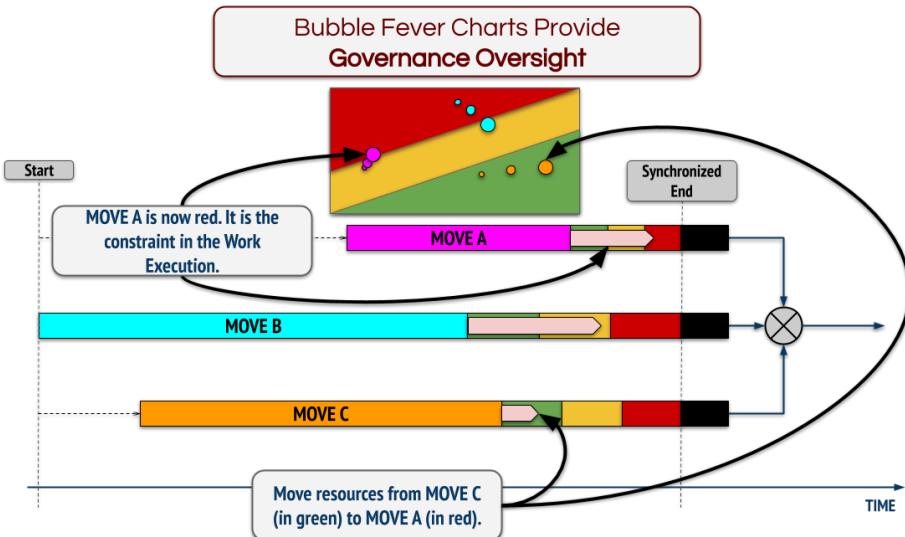


An animated GIF of this can be seen in Steve's "Visual Portfolio Management" blog post where the idea of a *Bubble Fever Chart* was first presented. See here:

<https://tameflow.com/blog/2014-11-25/visual-portfolio-management/>

In a *Bubble Fever Chart*, instead of representing a *MOVE* by plotting a line, we use a series of bubbles. We use only a limited number of bubbles for each series (i.e. *MOVE*) - in the diagram, we can see three bubbles for each series - in order not to clutter visualization. The bubbles in a series are drawn with increasing size, as they progress towards delivery (from left to right). The biggest bubble (the right-most one) is the data point of the current (or most recent) reporting period; and the decreasingly smaller bubbles to the left are the data points of the preceding reporting periods for that series.

The last *Bubble Fever Chart* frame above is representing the situation we had at the end of the previous chapter. If we take the last illustration of the preceding chapter, and add the latest *Bubble Fever Chart* to it, we would see the following:



Focused Governance

The *Bubble Fever Chart* gives us additional information over the visualization of the *Buffer Consumption*. It not only tells us that *MOVE A* is in the red zone, but also that it has been slowing down dramatically in the last three periods. It is visible because the pink bubbles of *MOVE A* seem to pile into each other, like cars in rush hour. The *Buffer Signal* is clearly readable too. *MOVE A* has just moved from the yellow zone and it is going to red zone, so some action is needed.

The cyan *MOVE B* is clearly progressing faster than *MOVE A*. The bubbles are more spread out and leave a longer trail. In fact, *MOVE B* is recovering, and as we can see, it is going from the red zone to yellow zone. It appears that the recovery will bring it back on track for a successful delivery. The fact that it was once in the red zone a few weeks ago yields no benefit to the decision makers today, nor should it trigger any discussions.

MOVE C is going very well. It is the fastest; the bubble trail is very long; and it is progressing toward early delivery.

By simply looking at this *Bubble Fever Chart*, the *Portfolio Manager* should make the decision to use the *Excess Capacity* - using fungible resources for real time labor liquidity decisions at work - of *MOVE C* to help *MOVE A* recover. (If the skills and knowledge are not available, then people from *MOVE C* could still use their *Excess Capacity* to try to learn the skills and acquire the knowledge needed to support *MOVE A*.)

The representation of the *Bubble Fever Chart* is vivid, and its interpretation immediate. The purpose is to move the bubbles from left to right as soon as possible, and whenever there might be conflicting priorities, only economic considerations that benefit the *Portfolio* as a whole should be used to decide where action should be focused (and what should be done).

This kind of visual representation - that makes us *stop talking and start acting* - can be generated in real time with the appropriate instrumentation and tools.

Many conventional management and steering meetings (status meetings, Scrum of Scrums, review meetings, etc.) will become redundant, and bypassed altogether using *Bubble Fever Charts*. This avoids a lot of coordination and transaction costs.

Instead, meetings might only be called when there are signs of trouble, and they will be focused precisely on the trouble spots. The approach will also give ideas about where help can come from, and prevents team territoriality from festering. The common agreement is about everybody collaborating and collectively taking the best actions that benefit the organization as a whole, rather than just a single team or project.

The sample *Bubble Fever Chart* would give top managers one simple chart to supervise all ongoing projects, and zoom in on specific problems. With this chart, *Executive Management* will understand at a glance that *MOVE A* is in dire straits. They could also see that *MOVE C* has ample margin, and hence decide to let *MOVE C*'s team members help *MOVE A*'s.

A most valuable consequence of this kind of visualization is that management need not be involved in the daily decision making. People on the ground can make the necessary decisions about constraints management (starting with the *Five Focusing Steps* and any other relevant action, like reallocating workforce liquidity). Top management needs to be involved only when the actions required fall outside of the workforce's own *Span of Control*, when they necessarily need to escalate issues to top management. And then top management will be able to "see" at a glance where the problems are and the context around them with all relevant information. And it won't be based on yesterday's news, once it is too late and already in escalation mode. The *Tameflow Approach* builds trust across the chain of command.

Of course, this is where conventional territoriality issues need to be overcome and replaced by the sharing of burden that needs a *Community of Trust* and *Unity of Purpose*. It is the essence of the story of Herbie. (See the *Prologue* to this book!)

To be even more effective, when there are hundreds or thousands of initiatives to handle, the charting tool should provide ways to filter out those elements that do not require attention.

The filtering that are most common to aid in automated decision-making are:

- Filter by "reddest" to find the trouble spot.
- Filter by "greenest" to find available resources.
- Filter by acceleration in the *Buffer Burn Rate*, to see if the speed is increasing (bubbles spreading apart) or decreasing (bubbles piling up).

Recall that all *MOVEs* will always be scheduled so as to arrive at a synchronized end date, with respect to a real or virtual integration point. With the above logic, we have a way to quickly zoom into the elements that are the main reason for keeping us from achieving that synchronized delivery.

We can see where *Bottlenecks* are forming (where the bubbles are piling up) before we run into problems, rather than detecting them after they have become visible queues of work (as we would do with *Column WIP Limits* - and only if we are lucky enough that those limits do not actually mask out the real queues). We can get an immediate feeling if those *Bottlenecks* are at risk of becoming the *Constraint* in the *Work Execution*, and act swiftly.

The Meaning of Red

It must be stressed again that if a *MOVE* goes into *red*, there is no reason to panic - unlike other *red-yellow-green* status reporting that are customary in conventional project management. Going into *red*

simply means that there is lots more negative variability than expected that is hitting the *MOVE*, and that action must be taken. It does not mean the *MOVE* is late. The *MOVE* is at risk of becoming late unless something is done. The *MOVE* will not be late until 100% of its buffer is consumed (what we earlier called “*going into the black*”).

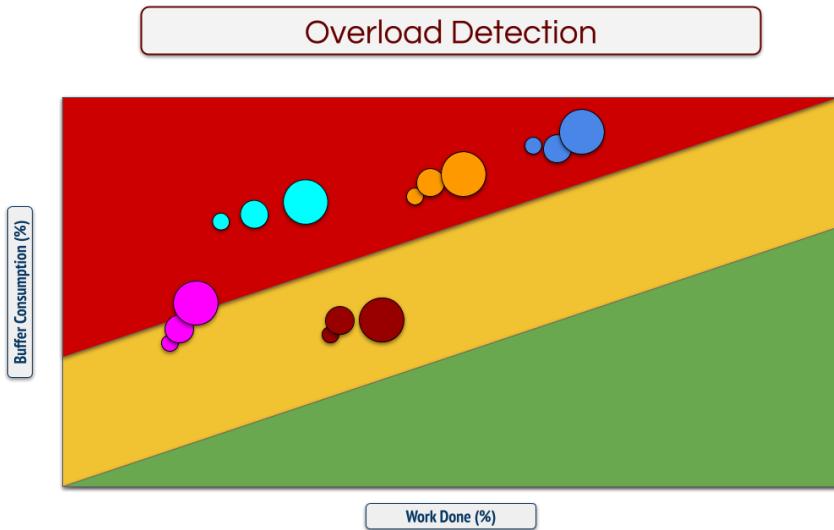
In the weekly or periodic reviews (typically when performing the *Full-Kitting* activities), it is the narrative that is told that will convey the real gravity of the situation. The *Product Owners* will tell their stories about the particular cases, and the gravity of those stories will be assessed, beyond the fact that they went into the *red zone*.

Note: Why would these reviews happen during *Full-Kitting* activities when *Full-Kitting* is typically used for planning and preparing future commitments? Simply because any *Full-Kitting* cannot be done properly while ignoring the current *Constraint*, and in particular the *Constraint* that has come into being in the very recent past - in other words any new *Constraint* in the *Work Execution*. So this reveals another facet of the *TameFlow Approach*. While we will diligently do reviews at the end of every *MOVE*, we will also engage in proactive monitoring - and perform *ad-hoc* reviews if need be - during the *Full-Kitting* activities. We will review any *MOVE* that demands attention due to its *Execution Management Signals*, and in particular in order to consider if it will impact the delivery *Capacity* and the preparation of the next steps that happen during *Full-Kitting*, in addition to, naturally, mitigating the situation for that particular *MOVE*. It is another instance of *Management by Exception*: we contemplate having reviews *on demand* when needed - we might decide to delay the start of new work when there are signals of disarray, so that we can focus on where the problems materialize, rather than mindlessly starting new work.

Overload Detection

One important benefit that the *Bubble Fever Chart* gives us is a visual way to detect when and if the entire system is overloaded. All one needs to do is to notice if the majority of *MOVEs* are in the red part of the chart or not.

For instance, look at this sample chart:



A *Bubble Fever Chart* like this one (where most items are in the red zone and none are in the green zone) is a loud descaling alarm, and requires immediate top management intervention.

Top managers would see that the entire system is overloaded, and most likely no *MOVE* will meet its expected due dates. The only solution is to descope - just limiting the amount of *Work in Process* is not sufficient. More drastic intervention is needed. There are too many problems hitting the *Work Execution*.

Note: Remember that we have a *Target-Scope* in a *MOVE*, and it can be made larger or smaller - as long as it is minimal and we are still able to deliver market value. At times, descaling a *MOVE* might require reaching out to our *Sphere of Influence* because we could need to renegotiate expectations with customers.

Some *MOVEs* will have to be stopped in their tracks and postponed - if not entirely cancelled - to make resources available and alleviate the situation of the most challenged *MOVEs*.

But the most obvious decision remains that no new work should be started when situations like this are witnessed. Adding *Work in Process* is the last thing we want to see.

In this phase, the prioritization techniques based on *Throughput Rate* learned in *Chapter 13 - Prioritization and Selection in PEST Environments* will become useful to decide which *MOVEs* to stop or kill, and which *MOVEs* will receive the freed up people.

In any case, the *Bubble Fever Chart* gives us a clear picture of relative criticalities of how the whole system is performing and allows us to focus decisions where they matter most. Top management will definitely appreciate being in the driver's seat, rather than being taken for a ride as is typical in most *PEST* environments.

Thinking is Still Required

If a *Bubble Fever Chart* has a majority of *MOVEs* in the red zone, then it is most *likely* a signal of system overload. It should appear once or twice, and then disappear once top management really addresses the issues. However, if the “redness” persists, it could also be an indication that the size of the *Buffers* are too small or the *MOVEs* are too big.

In this case, top management just has to change their expectations; and accept that the teams have to be given more time to do a proper job for that level of *Work Load*. The alternative of reducing *Work in Process* (which sometimes is hard especially when many things need to be ready by a deadline) is to take more time.

If overall *Work Load* cannot be reduced because of looming deadlines, then the only sensible lesson for the future is to start work *sooner* - and consequently making the *Buffers* larger.

The opposite is also true.

If a time period shows a majority of items in the green area, then it could be just a lucky transient period. If the “greenness” persists, we apply the same thinking in the opposite direction. The *Buffer* is probably too large for the *typical* kind of work the system has to perform. We can increase the *Work Load* if it makes sense. Though we must still remember that we have to *Postpone Commitment* of starting work in order to keep *Work in Process* under control. If it does not make sense to increase the *Work Load*, then we can reduce the size of the *Buffers*.

Note: When the effect of *Execution Management Signals* is experienced, it is natural to want to *tune* the *responsiveness* of the system. It can easily be done by reducing the size of the *Buffers*. However, in so doing, we must proceed with extreme caution. The system needs to be kept stable. Too much tinkering with *Buffer Sizes* can degenerate into *Tampering*. In an ideal situation, *Buffer* sizing should be automatic and based on statistical approaches, but lacking that, any rule-of-thumb heuristic that provides reasonably responsive actionable signals is good, and will be way better than any conventional “rear-view mirror” *Portfolio Management* approach.

Full-Kitting Revisited

The quality of work done during *Full Kitting* will impact how well the new nervous system will function. We do not want to interrupt work to avoid *Multitasking* and the accumulation of *Wait Time*. Our forecasts (or estimates) need to be reliable, because scheduling will be done around the *Constraint* in the *Work Flow* on the basis of the longest queue (expressed as *Flow Time* forecasts) that is formed in front of the virtual integration point.

Yet when the *Flow Managers* decide what gets scheduled next (even if the virtual queues of expected *Flow Time* forecasts in front of all the teams indicate that there *should* be available time slots and *Capacity*), they must not ignore the fact that “things happen.” It is possible that in the meantime, a significant *Bottleneck* or even a *Constraint* has surfaced in the *Work Execution*, and (depending on its severity), it might impact the scheduling of any further work.

In this sense, the *Full-Kitting* activity must be considered more as an *executive* function rather than a *planning* one. The scheduling and deployment of any new work into the *Work Flow* cannot be done while ignoring the contingent operating conditions of *Work Execution* at *that moment*.

The current state of *Work Execution* must always be considered first.

Remember the patient in the hospital metaphor? Any patient undergoing surgery right now must be dealt with before we operate on the next one. So if the current operation is taking longer than expected, we have to *postpone* starting the next one.

Likewise, if the state of *Work Execution* of a *MOVE* indicates that some critical delivery will be delayed, and become the *Constraint* in the *Work Execution* and jeopardize the delivery on the synchronized end date, then the synchronized end date must be moved forward and the planned start dates for the scheduled projects must be adjusted accordingly.

Similarly, at top management's weekly meeting (which is *ordinarily* called to resolve requests for ranking), any *Work Execution* problems being escalated must be addressed first.

Top management will have the "*patient in the hospital*" metaphor as a mental image, and will understand that their prompt responses will be fundamental for resolving the specific issues being escalated. Their mindfulness is also important in order to raise the odds that the future projects will be considered in relation to the evolving performance expectations. It is by having this performance-focused mindset - across the entire organization - that promises can be kept, and the competition outperformed.

Management by Exception - Limit Meetings

Meetings are one of the greatest drains of resources and psychological energy. The *TameFlow Approach* is deeply aware of this, and deliberate actions are taken to eliminate meetings altogether, or to alternatively decrease their frequency and duration as much as possible.

An expression was created with just that in mind: *Management by Exception*.

The *Kanban Method's Cadences* are no longer required as the full array of *Management by Exception's* tools provided by the *Tameflow Approach* aim at one objective that Agile has forgotten: *Let people work!*

The usage of *Bubble Fever Charts* can compress typical steering committee meetings from several hours to just 30-45 minutes, involve one third of the usual attendees, and limit detailed artefacts generation to only the projects in the red zone. Some meetings are obviously necessary, like the regular weekly *Full-Kitting* meetings or the *Request for Ranking* meetings as they are executive functions and operate better on a regular cadence. Yet many others - such as the daily rituals - are mostly unnecessary.

Management by Exception - Ad-hoc Meetings

The key notion in the *TameFlow Approach* is that unless a regular cadence is absolutely necessary - which typically occurs in "upstream" activities where work needs to be organized before it is started, like the *Full-Kitting* meetings - then a meeting will be required only on demand.

Specifically, meetings will be called for when there is a clear signal from the *Work Execution* that something needs attention. The events that trigger a meeting during *Work Execution* are primarily transitions between *Buffer Zones* and *Work Item Ageing* signals on individual boards.

This is not a rule cast in stone, but the expected default behaviour.

If anyone in the organization becomes aware of some relevant piece of information, a meeting should be arranged, even in the absence of a triggering signal - yet this should be the exception rather than the rule.

Effective and Focused Standup Meetings with the *TameFlow Approach*

Standup meetings in the *TameFlow Approach* are different than what is customary in Agile. These meetings are pre-scheduled with optionality. The teams will schedule recurring timeslots every day for their *Standup* meeting. Then, once the time comes, they look at the *Work Execution Signals*. If there are ageing items or new buffer conditions, then the *Standup* meeting is held.

If there are no signals, nobody is expected to show up, and the meeting is automatically cancelled, without even the need to notify the people. If meetings are needed, then the signals should set the agenda.

Naturally the *Standup* meeting can also be called on demand, should anyone discover new relevant information or major issues and problems that have not yet shown up as *Work Execution Signals*.

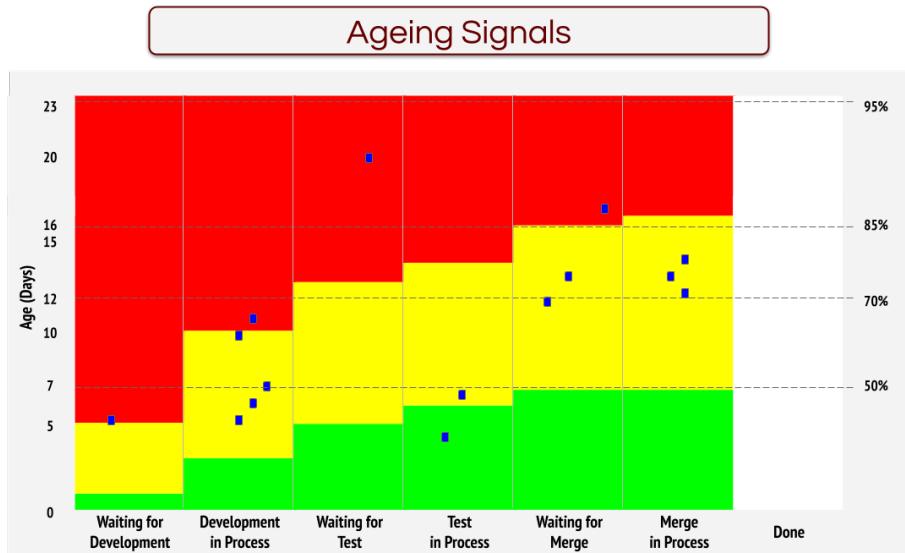
This way of having pre-scheduled *Standup* meetings confirmed only when there are relevant signals or called on demand is a lightweight protocol for assuring that the meetings are held only when there is a real reason, and not as a rote, time-wasting, daily ceremony. In fact, conventional Agile *Standup* meetings can be considered as a counterproductive practice. In large organizations, several thousand productive engineering man-hours can literally be wasted. Focus is the way to avoid that.

The engineering team's *Information Radiators* (screens, charts, whiteboards, etc.) with all the *Fever Charts* and *Ageing Charts* should always show updated information about the project - that is something that the *Flow Manager* (or *Scrum Masters*) should take care of in their daily routine. When there is a *Buffer Zone* penetration signal, then a standup meeting should be held at *the first opportune time*. Ideally a meeting should happen as soon as a signal is detected.

The teams are in touch with their work, and they will be the first to know when something is wrong. However, in the heat of the action, sometimes the significance of smaller events is ignored.

While we might be in a situation of not having any alarming information and no relevant *Buffer Signals*, another element that should trigger action are *Work Item Ageing Signals*. For this we can use a *Work Item Ageing Chart*. (For more information about these charts, see Daniel Vacanti's *Actionable Agile Metrics for Predictability* book.)

In the *Ageing Work in Process* chart below, we can see that one single *Work Item* is stuck in the *Waiting for Test* column. In this situation, ordinary *Kanban Boards* with *Column WIP Limits* would probably not raise any attention, since the rest of the work is regularly “flowing” through the board. Even the *Buffer Signals* might not be alarming.



That single item in the *Waiting for Test* column could be stuck there because some specific piece of equipment is missing. Maybe the *Full-Kitting* process missed that it had to be available in time for the testing of that *Work Item*. Or maybe the equipment was expected to be available, but something happened and it's now late.

The *Work Item* in the *Waiting for Test* has already way passed the 85% mark of due date probability for that stage of the *Work Process*. This should be escalated immediately to top management, in time for their next weekly meeting. If we were to find red items in more than one column, it could also be an early indicator of upcoming *Buffer Penetration* events - so always keep an eye on the *Buffer Status* as well.

In this case we must realize that even though there are no overall causes for alarm, the fact that one *Work Item* is stuck while the rest of the *Work Items* are flowing can lead to an unpleasant surprise at the end - or even block the delivery of the entire *MOVE!* And if this has cascading effects, it could become a threat for the orderly delivery of all other *MOVEs* at the scheduled common end date and integration point.

This is what we mean by *Leading Signals*. If interpreted correctly, they can allow us to take corrective action much sooner than when “*things happen*” - because we can realize they are about to happen, and we can act before they actually do.

Finally, while the *Buffer Consumption* is in the red zone, or when any single *Work Item* has gone beyond the acceptable ageing thresholds, then the standup meeting will be held on a regular daily basis until the issue has been resolved, or escalated for top management decision and action. It is then that *Non-Constraint* resources must be fungible to support staff liquidity in the system at all levels, as much as possible, and make good use of their *Excess Capacity*.

Note: If someone objects to the lack of opportunity for social interaction that might happen during the

daily standup, then learn from the Swedes and institute a morning and an afternoon *Fika* break. It is so significant, that while we do *not mention it formally*, we *do consider Fika as an essential part of the TameFlow Approach!* And leave your work on your desk. This happens among friends.

MOVE Reviews and Retrospectives

In *TameFlow*, reviews and retrospectives are not held regularly on a predetermined cadence. They are held only when significant *Events* happen. They could be held immediately, in an ad-hoc fashion, whenever a *Standup* meeting detects a major reason for calling them.

In general though, there will be a Review/Retropective whenever a *MOVE* is delivered.

If the *Buffer Signals* have been handled properly, and *Reason Logs* been kept regarding plausible causes of *Buffer Penetration* or *Work Item Ageing* episodes, then most times such retrospectives will be dedicated to performing *Root Cause Analysis* (as explained in the *Hyper-productive Knowledge Work Performance* book), and they will be more factual rather than based on anecdotes or opinions of the participants.

Consequently, such meetings will be shorter. There will be more agreement on causes and effects, and on the actions to undertake next.

Note: Many of the activities that occur in the conventional *Sprint* planning, review and refinement meetings happen on a more continuous basis in the ongoing *Executive Full-Kitting* activity in *TameFlow*.

Informational Flow

There is a lot of content on *Informational Flow* in the *TameFlow Approach* that builds upon these topics and further develops the patterns that create the organizational nervous systems so that it is attuned to performance. But that is a topic for another book. What we have seen so far is setting the basis for effective *Operational Flow*, which is the foundation on top of which *Business Agility* can be built.

Notes on Cadences

Like the flow of a river in nature or the earth's orbit around the sun, progress goes uninterrupted and unattended. The laws of nature will let things evolve effortlessly without intervention.

Now imagine a 3 year project or Operations "flowing" in that type of system. Well, *Common Cause Variation* and *Special Cause Variation* would take us for a costly ride. In the context of Agile flow based systems like Kanban, we cannot simply let things go. We need heartbeats to stop, check the status of progress, inspect, and adapt.

Such is the nature and purpose of *Cadences*.

For example Scrum has plenty of heartbeats the most drastic of which, at the end of a sprint, pulls the plug on any accumulation of variability, only to restart anew - basically cancelling any information contained in that variability!

Note: The negative impact of the anti-pattern of “resetting the variability” on a *Cadence* might easily escape Agilists’ attention. If some unfavorable variation is hitting a *Scrum* team, they will not be concerned about its implications. Why? Because any unfinished stories are simply spilled over into the next *Sprint*. When that *Sprint* is planned, velocity is adjusted to take into consideration the slow down. This pattern can repeat itself over and over again, one *Sprint* after another. There is no instrumentation or capability of detecting the aggregate impact of these episodes, unlike the *TameFlow Approach* where *Work Execution Signals* will effectively trigger ad-hoc retrospections - which through the deeper insights gathered from the compilation of *Reason Logs*, *Frequency Analysis* and *Root Cause Analysis* will help detecting and managing *Common Cause Variation*. The *Sprint Retrospectives* just don’t have sensitivity for all the information they are losing by “resetting the buffers” after every iteration. As for Kanban, the cadenced reviews have no *Buffer Management* equivalent to trap and handle variability in such a sophisticated way. Both are foregoing one of the greatest drivers toward superior performance: visibility over *Common Cause Variation*.

There are many reasons why *Cadences* are used with the *Kanban Method* including to drive evolutionary change and achieve continuous improvement. While there is merit to the former compared to all that is offered in the market in terms of Agile, we have seen throughout this book why this can be misleading.

The Kanban *Cadences* never aim directly at the system *Constraints* inhibiting any kind of actual continuous improvement. The Kanban Maturity Model’s (KMM) consideration of *Constraints Management*, *Throughput* and *Flow Efficiency* is weak.

There are other reasons for the use of *Cadences*. Here are the most noteworthy:

1. Kanban *Cadences* aim at mitigating the **compounding of variability due to the interdependency of knowledge-work processes**. For example, if one team has to wait for another team to finish before it can start, the former is at the mercy of whatever variation happens to the latter, to which its own variation will be added. Scrum has a remedy for this as it “pulls the plug” and resets at the end of each *Sprint*, deliberately controlling how much variation is allowed to enter the system (but also, as we said, foregoing the opportunity of acting on the information borne by that cancelled variation). Kanban *Cadences* are heartbeats meant to mitigate the compounding of such variability due to process interdependency.

Note: The *Theory of Constraints* controls variation by subordinating all interdependencies to the *Constraint* of the system, and then managing according to the performance of the *Constraint* via *Work Execution Signals*. It does not sidestep the detection of variation through rituals. Variation is not cancelled through resetting cadenced actions, nor hindered artificially with *Column WIP Limits*, and hence can be used to steer attention and action to where it is really needed.

2. The positive consequences and indisputable benefits of the Kanban *Cadences* reside in the **decoupling of input with output**, thereby permitting commitment and reliability to be reached with observation and measurement as opposed to planning and estimating. The *Tameflow Approach* is slightly different. We do not think that adequate planning is the problem, even though metrics and probabilistic forecasting are equally used in preemptive planning. The real problem is in acting on every minute deviation of such plans, or every small false positive signal that stems from exceeding *Column WIP Limits*. The *Tameflow Approach* is unique as we focus on *Work Execution* (not the plan, not the deviation, not the breaking of *Column WIP* limits) with the advanced managerial techniques and leading indicators from the *Theory of Constraints* and *Throughput Accounting*.
3. **Negative (Balancing) Feedback loops:** Kanban *Cadences* also appropriately serve as self-regulating systems. The *TameFlow Approach* creates a whole new array of feedback loops which are triggered in real time by the *Work Execution Signals*, resulting in much tighter and reactive loops supporting higher levels of overall performance.
4. **Scaling:** The quintessential objective for Kanban is to scale. The *Tameflow Approach* to scaling is counterintuitive. While it has been illustrated in practice in the *PEST* sections of this book, the essence is the adoption of *scaleless* organization-wide *Mental Models* supporting decision-making at all levels, striving to create *Unity of Purpose* and a *Community of Trust* in the pursuit of a common organizational *Goal*. Emphasis is on creating the conditions for the alignment of all such autonomous decisions (at all levels) rather than compliance to processes, guides, frameworks, maturity levels, ceremonies, practices, principles or values - all factors that are the result of using *flawed Mental Models*, rather than using *Mental Models* as the primary driver of all decision-making.
5. **Triage:** What do we do now, later or never is a recurring theme in most *Cadences* to manage options. This is addressed in more powerful ways with the *Full-Kitting* meetings where triaging, prioritization and sequencing criteria are based on the *current knowledge* about the *Constraint* in the *Work Flow*, *Work Process* or *Work Execution*.
6. **Reliable work:** Heartbeats are the foundation of predictable delivery. The *Kanban Method* uses *Cadences* for this. It is linked to the accumulation of variability during the passage of time. From a *TameFlow* perspective, heartbeats are overrated. What matters is - to use a sport's term - to keep your eyes on the ball. Regular meetings tend to become ceremonial and kill off motivation, energy and focus. The *TameFlow Approach* will insist on having meetings whenever they are necessary - and only when they are necessary. Furthermore, such meetings are never timeboxed (as is customary in most other approaches). Meetings focus on achieving objectives, outcomes, decisions and action, and they end as soon as those have been achieved. The combination of the target scope *MOVEs*, *Full-Kitting*, and *Work Execution Signals* (*Work Item Ageing Signals*, *Buffer Signals*, and *Bubble Fever Charts*) provide very lightweight, fast and efficient ways to know when meetings should be called. Clear objectives, targets and *Management by Exception* are the *TameFlow Approach*'s simple replacement of the process overload - and consequential process fatigue - of other approaches.

Specific Kanban Cadences

This section describes each traditional *Kanban Cadence* in detail with the further perspective that the *Tameflow Approach* brings as it relates to the content of this book. Note that each *Cadence* does not require a meeting of its own and that the essence of each can be incorporated into current practices.

Strategy Review : The *Tameflow Approach* is all about *Flow*, especially at the strategic level. Modern techniques, like the *Incremental Funding Method* and *Beyond Budgeting* are topics that promote *Flow* and create value rather than adhering to prescribed *Cadences* in an exercise of conformity. The *Theory of Constraints* and *Throughput Accounting* of Dr. Goldratt, are applied throughout the *TameFlow Approach*

(with other elements from *Pattern Theory* to *Organizational Psychology*) have no equal in the *Kanban Method*, and are an essential part of strategic decision-making.

Operational Review: Also known as the systems of systems review. Be very prepared is the warning served when we are nearing the *Operational Review* deadline. All hands on deck are required to gather metrics and everybody goes to the review. The use of *Management by Exception* is simply unknown and ignored. When we can avail us of leading indicators - and there are quite a few - and their signals are green or yellow, why waste valuable cycles? Let people work! It is also here that performance is said to be evaluated. It is also here that the use of *Throughput Rates* in the *Tameflow Approach* supersedes all metrics of the *Kanban Method* - which doesn't have a metric to gauge *Financial Performance*.

Risk Review: It is a forward looking review anticipating what could go wrong. This is where the *Tameflow Approach* sets itself apart from the *Kanban Method*. Contrary to the *post-facto Event Driven Risk Management* paradigm of the *Kanban Method*, the *Tameflow Approach* :

- Includes *Explicit Risk Management* in its model.
- Does not ignore *Common Cause Variation* - the most powerful risk management and mitigation tool at the disposal of knowledge-work!
- Offers a complete array of *Leading Indicators* of oncoming risk materialisation, enabling the *Management by Exception* approach.

Delivery Planning : In the *TameFlow Approach*, delivery planning is part and parcel of *Full-Kitting*. Remember that *Full-Kitting* is preparing for the work to go through the *Work Flow* in an uninterrupted manner - all the way to final delivery. Typically, this is done when defining a *MOVE*. The actual delivery is nothing less than a mere phase of the *Work Flow*, and is managed consistently with attention to *Work Execution Signals* and *Management by Exception*.

Service Delivery Review: In the *Kanban Method*, each service is reviewed with its *Flow Metrics* and adherence to *Service-Level Agreement* objectives. In the *TameFlow Approach*, any metric or criteria that define customer expectations are part of the definition of a *MOVE*. Remember that one of the objectives of a *MOVE* is to create an outcome and/or a value that is acceptable to a paying customer

Kanban Daily Meeting: This subject was addressed earlier in this chapter. *Work Execution Signals* are the preferred choice to support *Management by Exception*. Scheduled daily meetings are a waste of time when there is nothing to discuss. In order to avoid chaos, it is obvious that the timeslot and necessary resources be reserved to hold such meetings. The benefits in the ritual of going through a *Kanban Board* from right to left are weak compared to the insight that *Work Item Ageing Signals* and *Buffer Signals* deliver instantly.

Replenishment Meeting: Suggested as a 30 minute meeting in the *Kanban Method*, it is for a different granularity of incoming work. In that regard, the *Tameflow Approach* is more in line with *Upstream Kanban*. It is therefore replaced by *Full-Kitting*, *MOVEs* and *DBR Scheduling*, and it involves participants from all levels of the organization, up to the CEO if needed (as described in the section on *Operational Full-Kitting* at the end of *Chapter 18 - Full-Kitting as Ongoing Executive Activity*). Note that via such *Full-Kitting*, the organization effectively engages in *Continuous Replenishment* that is triggered by the current *Constraint* - via *DBR Scheduling* - and how the *Constraint* is affecting the overall delivery capacity of the organization. Again, it is based on *Work Execution Signals* and the *Mental Models* that allow us to appreciate how overall performance is based on a profound understanding of the entire system.

A Vocabulary for Thinking about Constraints

Constraint Management needs a meaningful vocabulary so that we can effectively reason about it. It is important to develop our language to reflect our understanding.

What differentiates a *Bottleneck* from a *Constraint* is simple: the *Constraint* can be actively managed, while trying to manage *Bottlenecks* will give you the entertainment of a whack-a-mole game.

But the *Constraint* can appear in different places at different times. No matter where and when it appears, we must always be in a position to manage it. We are now able to distinguish between three types of constraints

Here is a short summary of the three *Constraints*.

Constraint in the Work Flow

- The **Jungle** in the *Jungle, Jeep, and Journey* metaphor.
- “*Things that we see coming toward us.*”
- The business domain; the market/customer demand.
- Within our *Sphere of Influence*.
- VUCA
- Like a map which guides us through our *Journey*.
- How to detect it: Longest *Work Load* queue in front of a team.
- Cure: *Portfolio Management* board with *DBR Scheduling* orchestrated by the *Constraint* in the *Work Process* (see below), and ongoing executive *Full-Kitting* (see *Chapter 18 - Full-Kitting as Ongoing Executive Activity*).
- One of your teams is this *Constraint*.
- Caused mostly by *Special Cause Variation* and to a lesser extent by *Common Cause Variation*.

Constraint in the Work Process

- The **Jeep** in the *Jungle, Jeep, and Journey* metaphor.
- “*How we do things around here.*”
- The “*Herbie*” team.
- Within our *Span of Control*.
- How to detect it: the column with the highest average *Flow Time* of our process on a *TameFlow Board*.
- Cure: *DBR Board* or *Throughput Management Board* with *DBR Scheduling* orchestrated by the *Constraint* column, plus *Reason Logs, Frequency Analysis* and *Root Cause Analysis*.
- Caused mostly by *Common Cause Variation*.
- Where the most worthy improvement opportunities are found.

Constraint in the Work Execution

- The **Journey** in the *Jungle, Jeep and Journey* metaphor.
- “*Things that happen.*”
- Moving through the *Jungle* and finding surprises: it is the project “execution” meeting the unexpected.
- Beyond our *Sphere of Control*.
- We do not know what to expect; we are not in control of it; we cannot influence it.
- How to detect: *Execution Management Signals* (as explained in this chapter).
- Cure: Use *Buffer Signals* to trigger “*Inspect and Adapt*”. And use *Buffer Management* to “*Sense and Respond*.” Enhance with *Reason Logs, Frequency Analysis* and *Root Cause Analysis*.
- Caused almost entirely by *Special Cause Variation*.

Takeaways

What does “*Governance*” mean? One thing is for sure, it leads to and creates obscurantism for those who are excluded from the process! It is often forced externally by some “*Rules*” and always materializes itself as a top down imposition set forth in the name of procedural conformance to put an end to ailments brought about by the failures of our organisational structure.

But does it and has it ever? Our favorite description of governance is “*Who gets their say, who gets their way.*”

We can smell the power at play!

Still governance is a catchy thing. It paves the way to a quest for transparency and fairness when it is triggered voluntarily by a company.

But where is the equalizer in a leveled playing field of procedural imposition and conformance, where the small peons only have their say and the big chiefs always have it their way? It all lies in relative scales. It all lies in front of us in this book.

This book is an ode to governance. Small or big, we will wear our true colors - green, yellow or red - and get the attention and care we require. This is procedural justice. This is fair.

Don't leave Herbie alone in the woods!

Tom Cagley says...

The only higher praise I have for a book than “it is useful” is “it is very useful.” Steve Tendon and Daniel Doiron provide the readers with an actionable path to improving flow.

“Tame your Work Flow” is a book that is on my must own, must read and must carry when helping clients improve value delivery.

Tom Cagley, Author of *“Mastering Software Project Management Best Practices, Tools and Techniques”* and Editor of the *Software Process and Measurement Cast (SPaMCAST)*.

PART 7—Igniting High Performance

In this last part of the book we will provide actionable *hints, tricks and tips* on how to get started with all the ideas we have learned thus far.

These hints, tricks and tips are called *Patterns*. We will examine twenty-two such *Patterns*. Every one will give us something to do and to aim for. (These *Patterns* are part of a larger undertaking, but for the purpose of just getting started, the intricacies of *Pattern Theory* are beyond what we need.)

These *Patterns* will provide guidance on what to do (in the form of actionable *Assignments*) and what outcomes to expect (in the form of *Expected Effects*), so that we can transform all the theories and *Mental Models* covered thus far into practice.

Once we realize these twenty-two *Patterns*, we will be on the way to becoming a more productive organization!

Joseph Hurtado Says...

"Tame your Work Flow" is an unusual book, it manages to push the borders of Kanban towards the world of business-wide improvements, inspired by Goldratt's Theory of Constraints. Steve Tendon and Daniel Doiron deliver an enjoyable, innovative approach to improve any business area, or even whole companies.

Unlike other pre-packaged approaches, theirs is a goal driven method, that is sure to improve any company.

As an Agile Lean Transformation Consultant with over 15 years in the industry using Kanban, Lean, Scrum, and SAFe I wholeheartedly recommend it. It's an enjoyable, original, and enlightening book.

A must read!

Joseph Hurtado, Software Engineering Manager, Enterprise Agile Coach, Tech. Program Manager, SAFe SPC / Scrum CSM / Kanban Trainer

21—Patterns to Get Started!

While the previous chapters might provide the *Mental Models* to understand how to organize and manage knowledge-work at scale, in a PEST environment, it is one thing to have a conceptual understanding, and quite another to actually translate these ideas into practice.

Now, many will be asking: “*How can I do it? How do I get started?*”

We will try to answer those questions in this chapter, assuming that we have a genuine motivation to improve our organization.

The market seems to be crying for canned, cookie cutter solutions, and gerrymandered “best practices” that can be simply cut and paste - with the expectation that improvements will be witnessed overnight. Some stem from behavioural methods sustained with faith-based arguments - even belief-based actions - with no foundation in logic nor science. Others are founded on authoritative decrees of methodological maximalists - equally with little foundation in logic or science.

Unfortunately, cookie cutters, beliefs and authorities will not help us in the field, with *our* business and *our* problems. As we can read in the first couple of chapters of the *Hyper-Productive Knowledge Work Performance* book, the methods and practices that make one organization perform better than others cannot be easily replicated.

A Patterns Based Approach

At most, we can relate to such common elements in terms of *Alexandrian Patterns*.



Alexandrian Pattern: A solutions to a problem in a context.

We can try to recreate the scenarios wherein the presence of a number of known *Patterns* will generate the conditions of superior performance. It is not possible to institutionalize a cookbook of recipes that will magically transform a sluggish organization into an outstanding one. Instead, by reasoning in terms of generative and non-prescriptive *Patterns*, we will have adequate direction and sufficient flexibility to make them applicable to our own particular context.

Note: The patterns included in this chapter serve as illustrations to get us started. They do not meet the rigorousness of *Alexandrian Patterns*. While we will use the term “*Pattern*” extensively in this chapter, we might be better advised to consider them as *Proto-Patterns* or *Candidate Patterns*, because here we do not elaborate further on whether or not they present all the characteristics of a true *Alexandrian Pattern*, and if they can be part of a *Pattern Language*. However, the “*Patterns*” presented hereafter are certainly recognizable and actionable. They can serve as an approach to experiment and observe what the results are in our environment. They can give form; create a new vocabulary at work; and lead us to action. This is their intended purpose.

We do not want to have a laundry list approach because the risk is that we end up doing face value exercises only to check all the boxes and proclaim that we are compliant, mature, agile (or whatever) just by conforming to some imaginary standards.

Besides, if there were a standard, there would be no differentiation in performance to strive for, and we would just be drawn to the mediocrity of the average.

Since we do not believe in magic recipes, in this chapter we will suggest just *one* possible way - primarily as a means of illustration - by which an organization *could* be brought over to a state of Flow.

This is by no means a magic recipe or a framework. It is just an illustration of how things might possibly unfold in an organization by taking into account what we have learned in the previous chapters. Also, in the following pages we will get acquainted with other notions (which would prepare us to understand the *TameFlow Patterns*) that are not covered at all in this book, but that play a key role in how the *TameFlow Approach* is actually rolled out in an organization.

Baby Steps toward Improvement

In the next sections we will suggest actions that might involve us personally, our team, or even the CEO and all the management in the layers in between. Notice that we will express this as an actionable *Assignment* followed by a description of some *Expected Effects*.



Assignment: This is where the theory turns into practice and action: we will have something *very specific* to do.

The *Mental Models* studied so far will give us all the justification we need to understand why the assignment makes sense, and frame the action in relation to the broader picture of creating a high-performance organization.



Expected Effect: The execution of an *Assignment* will necessarily produce some tangible change. In virtue of our *Mental Models* we will hypothesize a certain result. We will explicitly spell out the result in terms of one or more *Expected Effects*. *Expected Effects* need to be detected categorically either through unequivocal *observation* or through effective *measurement*.

What we are doing here - even though we have hardly even begun scratching the surface of the topic - is aiming to apply some *TameFlow Patterns*. These *Patterns* all suggest that we *do something*, and then to *observe some results* as a consequence of our actions.

Note: This way of approaching the evolution is somewhat reminiscent of *OKRs* (*Objectives* and *Key Results*). The *Assignment* can be thought of an expression of an *Objective*, and the *Expected Effect* the equivalent of a *Key Result*. If our organization is already accustomed to using *OKRs* then there is a high degree of synergy with the *TameFlow Patterns*. The technical difference is a matter of topology: *OKRs* are organized in a hierarchical tree, while *TameFlow Patterns* exist in a network of interconnectedness. Furthermore, if we come from a *Theory of Constraints* background, we will see a clear correspondence

with the *Strategy and Tactics (S&T) Trees*. Even in this case the difference is in the topology: *S&T Trees* are organized hierarchically (as implied by their name), while *TameFlow Patterns* exist in a network. The *TameFlow Patterns* are more profound, flexible and powerful than either *OKRs* or *S&T Trees*.

Examples of what we might expect:



C-Level Patterns:

- The real capacities of company divisions, units and teams will be understood and used for managerial decision-making.
- *Throughput Rates* can be used to support investment decisions, make-or-buy scenarios or divestiture from markets.
- The company becomes a *One-metric Company* by knowing the *Throughput Rate* of its systems.
- The driving concepts and management vocabulary change.



Manager Level Patterns:

- Swift and unanimous decision-making will become the norm.
- Operations become more reliable.
- Promises can be made and kept.
- Trust increases at all levels - even toward external parties (clients and suppliers).



Knowledge-Worker Level Patterns:

- Morale improves.
- Slack (*Excess Capacity*) is accepted as a positive byproduct of a hyper performing organisation.
- Overtime will be an exception rather than the rule.
- Cooperation among co-workers increases.
- Territoriality is reduced.
- Time for skill advancement is created and recognized as being valuable.

We also want to identify *specifically* what to do (the *Assignment*) and to be able to discern whether or not we achieved what we expected (the *Expected Effect*).

Once the *Expected Effects* are really detected through *observation* and/or *measurement*, then that specific *Pattern* will be considered as resolved, and we can focus on to the next *Pattern*.



Pattern Resolution: The *Expected Effect* is present (observed and/or measured) as a consequence of the actions performed during the *Assignment*.

Until the time that the *Expected Effect* is present, different courses of actions might be experimented.

Note: If the *Pattern* cannot be resolved (which is extremely rare), then the learning outcomes will be incorporated into a new, evolved *Pattern* that will drive the actions in a different direction, until a resolution is effectively found.

It's important that we *will have one - and only one - change initiative in process at any given time*. Hence the title of this section: *Baby Steps* - we will take only one step at a time!

Unless we are able to attain the *Expected Effects*, we will not progress with the next change. We do not want to be caught in a situation where multiple changes are happening at the same time, and because then we do not know which ones are succeeding, failing or interfering with one another.

We also want to avoid the dreaded *change fatigue*, and give joy to those involved to celebrate partial accomplishments, as they happen, rather than having to work for months on monumental transformation programs, and then not even know if they achieved something - if at all.

Pattern Network Traversal is not Stage-Gated Project Management

Since the condition of meeting the *Expected Effects* is effectively a requirement to proceed with the next *Pattern*, we might assume that this is some sort of old fashioned stage-gated, milestone driven project management scheme.

It is not the case.

While the impression might be one of sequential linearity, if we study the *TameFlow Patterns*, we will discover that they exist in a network of interconnections between themselves, and that there are connective rules which determine which *Patterns* can be combined.

The presence of a *Pattern* determines an impact, which in turn can establish the context for yet another *Pattern* to come into existence.

We want to identify *Expected Effects* in order for certain desirable conditions and contexts to come into existence; and therefore make it possible to pursue and apply further desirable *Patterns*.

By knowing which of the organizational performance *Patterns* (and which combinations of these) are worthwhile, we can move along this network, in the direction of well identified connections.

By the end, we will have resolved and realized a *collection of performance Patterns*.

The particular selected path of interconnectedness that we choose to traverse will appear as a linear sequence, but it must be understood that is just *one* path out of many that are possible in the *Pattern Network* we can describe via a *Pattern Language*.

Pay particular attention to the fact that in the network traversal path we are concerned about has been selected with a deliberate intent. It is not a random walk.

Any *Pattern* can have multiple exits, but we deliberately choose one. This has nothing to do with *Gantt chart* stage-gated linear project management! Note that in more complex scenarios, we might want to effectively realize a networked *Topology of Patterns*, but that is not what we go after when we just want to *get started!*

Actionable Use of Metrics

Many of the *Patterns* will have *Expected Effects* that are not only observable, but also *measurable*. In these instances, the *Metrics* will reveal how much the *Pattern* resolution is effectively contributing to the overall initiative and improvement objectives.

The whole approach discussed in this book has a very strong focus on such *Metrics*, and in particular on *actionable Metrics* that can be used to make decisions - rather than just reporting on a past status.

Iterative Process of Ongoing Improvement

By focusing on a single *Pattern*-based change at a time, we can act according to this structured approach:

1. Observe the current situation of the organization.
2. Identify the underlying *Patterns*.
3. Learn about the *Mental Models* and *Thinking Processes* needed to understand why the *Patterns* happen and how they can be resolved.
4. Develop an action plan to resolve the *Patterns*.
5. Establish what the *Expected Effects* are; how they can be observed; and what *Metrics* can be used (if any).
6. Execute the activities to resolve the *Pattern*.
7. Observe the outcomes of the activities; validate the *Expected Effects* and/or *Metrics*; and reflect on the next steps.

If the *Expected Effects/Metrics* cannot be observed/measured in Step 7, then focus will go back to Step 4, and the iterative process is repeated until the *Pattern* is fully resolved. Occasionally, the resolution will be out of reach, then focus will return to Step 1 and learnings will be applied to a new direction of resolution. This means that we will pursue a completely new *Pattern* (with a new *Assignment* and new *Expected Effects*); or, in more advanced scenarios, we will move along a new path in the *Pattern Network* traversal that starts off from the same *Context* (in other words, the *Assignment* remains the same, but we look for different *Expected Effects*.)

If the *Expected Effects* or *Metrics* are effectively observed or measured in Step 7, then the process will be repeated, but this time to tackle the next identified *Pattern*.

After a series of *Pattern* resolutions, observable or measurable improvements will be seen, and eventually the improvements will conjointly manifest themselves as the primary measure of success!

This is a process of iterative ongoing improvement based on scientific observation. Once the primary measure of success is achieved, and the entire programme is declared as concluded, the company will not only have achieved the desired improvement, but will also have learned how to *apply a patterns-based methodology to keep on improving* thereafter - autonomously and without further support. We are in the domain of double-loop learning, whereby the entire organization becomes a *Learning Organization* and can constantly adapt to new conditions - with its own forces and no external help.

Pattern 1 - Leave No One Behind

The story of Herbie had one extremely important lesson: no member of the troop was left behind. They all arrived at the base camp together. Likewise, when seeking to achieve *Flow*, no one can be left behind.

The entire organization, from the CEO to the latest junior hire must be included. The *Mental Models* that provide the operational decision-making criteria of where to *focus* and what to *do next* must be shared throughout the entire organisation.

The *CEO (and all C-level executives) must be involved*, because *management attention* is one of the biggest constraints in an organization. C-levels do not have time to waste, but above all, they cannot afford to make the *wrong decisions* - or to allow the organization to choose the wrong paths by not having the right information to make the right decisions, at any time.

As decision-makers, top managers are ultimately making the most important choices. When applying the *TameFlow Patterns* we are definitely affecting the *Informational Flow* of the organization, making sure that the right information is available to the right people at the right time. So the top decision-makers need to be an integral part of the collective rewiring of the organization's nervous system. Only then will they be able to make the right decisions at the right time, and allow the organization to *change direction at high speed*, thus achieving *Business Agility*.

Assignment

If we are the CEO, then we are already onboard. Congratulations!

If we are not the CEO and are in charge of bringing our team, division, unit, or entire organization to *Flow*, our first assignment is to make sure our CEO is onboard.

This is a critical success factor. Without the active participation of the head and the entire hierarchy our efforts will be in vain. Do not go any further unless this is acquired.

Expected Effects

The CEO must commit to be an active participant. For instance, the CEO could agree to take part in an initial *Goal Setting* workshop; and then might commit to dedicate up to 30 minutes for review and feedback at the end of every week or every major event. The CEO understands that this is not only for our team, but is the foundation for improving the performance of the entire business - which just happens to start with our team - as we will see in the next section.

Pattern 2 - One Slice at a Time

The *TameFlow Approach* aims at significantly increasing the performance of an *entire organization*. However, *one must start somewhere*.

There are many approaches used in business. On the one hand there is *executive coaching* which is targeted at top management. On the other hand, typically with *Agile* transformations or initiatives, attention is on *all teams* in an organization or division. Then there are approaches that roll out changes across the entire organization all at once ubiquitously, like *Lean*, *TQM* or even classical *Theory of Constraints*.

All of these approaches have advantages and disadvantages. Some are disconnected between the upper and lower echelons. Others require a significant amount of time and effort to affect the whole organization and might even take years to complete. Some are perceived as impositions and do not get the buy-in and engagement of all people involved, whether at the top, middle or bottom of the organization. Therefore - with these approaches - something will always be left out (at the top or at the bottom). It will also give rise to blaming and finger-pointing when the initiatives fail or appear to fail.

Note: It is worth repeating. The ultimate intent of the *TameFlow Approach* is to create a *Unity of Purpose* that moves the entire organization, from top to bottom. The organization as a whole needs to be on board, and nobody left behind. The *TameFlow Approach* also builds on the concept of *Inspired Leadership*, and the CEO must be the first *Inspired Leader* to act in that role. (Consider, however, that in the *TameFlow Approach*, leadership is an *activity* that anyone can perform. It is not a role exclusively reserved for a top manager.)

One suggested way to get going is to start with *one team and all intermediary managers up to and including the CEO*, that gels together. The case for starting with a *vertical slice* of the whole organization is that in this way the organization's entire "nervous system" - from top to bottom - can be rewired to convey the *Mental Models*, ideas, and information that are needed to support superior performance. Many of the activities, during the course of our suggested patterns, will address the *Informational Flow*, especially the one that moves information and calls for decision-making across the organization.

Once a first *vertical slice* of the organization has acquired the *Mental Models*, mindsets and attitudes of the *TameFlow Approach*, it becomes much easier and much faster to roll out the approach horizontally, one slice at a time. The experience becomes almost self-propagating horizontally and organically, because the earlier slices and their successes will inspire the new ones.

Furthermore, the subsequent iterations will move much faster, because by that time, top management will have learnt what the *TameFlow Approach* represents to them, and will be able to support the transformation in a much more mindful and energetic way. Other slices will be able to learn from the experience of the earlier ones.

All in all, the whole organizational transformation happens much faster than with other approaches – while remaining successful, and reducing the odds of failure. The speed of adoption can be astounding, simply because there is full management buy-in and active support, and all actions reinforce one another in virtue of the developing *Unity of Purpose*.

The approach by vertical slice is also a wise choice as top-down approaches are always confronted with friction, while bottom-up initiatives will flounder because of waning momentum.

Assignment

Pick one team with which to start using the *TameFlow Approach*. To keep it simple while we are starting, the team might run a single *Kanban Board*.

Ensure that the CEO and all intermediary managers along the reporting line are well informed about the initiative, and – above all – involved.

Expected Effect

Personal commitment from the CEO to work with the team and all intermediary managers – and, likewise, personal commitment from all these individuals to work together.

Pattern 3 - Set The Goal

The founding book of the *Theory of Constraints* by Dr. Goldratt was entitled *The Goal* - and for a very good motive. Unless a very clear goal can be articulated to which everybody in the organization can subscribe to, there will be countless, unwarranted *conflicts* which will drain the organization of its focus, energy and effectiveness.

Without a clear goal, superior organizational performance will simply never happen.

With the *TameFlow Approach*, we are invited to start by developing a *Goal Tree*. This is an interactive and collaborative exercise. However, since there are so many problems that are habitual across organizations, we can devise a generic *Goal Tree*, to have some common ground to contemplate, and to see how the components of the *Goal Tree* justify pursuing many of the *Assignments* and *Expected Effects* that we are discovering here.



Goal Tree: A *Goal Tree* is a logic tree which connects the organizational *Goal* to subordinate *Critical Success Factors* and to further subordinate *Necessary Conditions*.

Every element of the *Goal Tree* is preferably expressed with a simple subject-noun sentence.

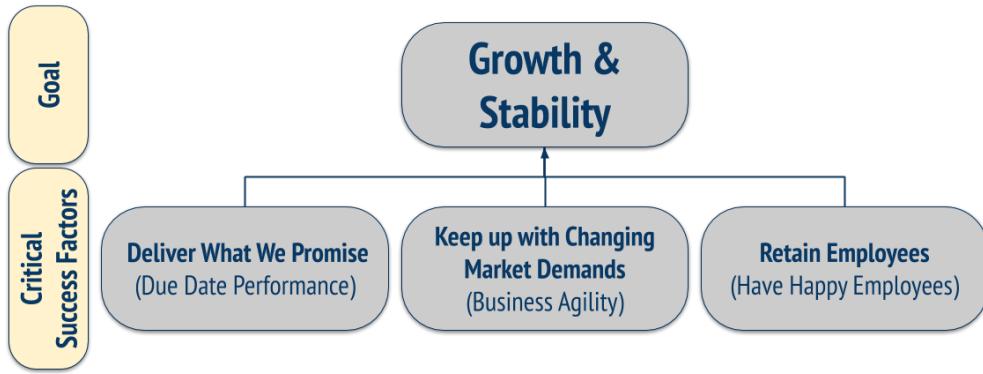
A *Necessity Logic* connection is represented with an arrow pointing from the *subordinate* elements to *superior* elements.

The *Necessity Logic* connections can be read out aloud with the template:

In order to <superior sentence> we must <subordinate sentence>.

The Goal

While we will have to develop our own *Goal Tree*, it can be useful to refer to a generic one. What can be a generic goal that all companies would subscribe to?



All companies are concerned about achieving *growth*. They want more profits, more market share, more customers, etc. Most companies need to maintain *stability*. If growth happens too quickly, the company will be unable to keep up and collapse under the weight it needs to carry.

So achieving *growth* while maintaining *stability* are acceptable shared goals for most of us.

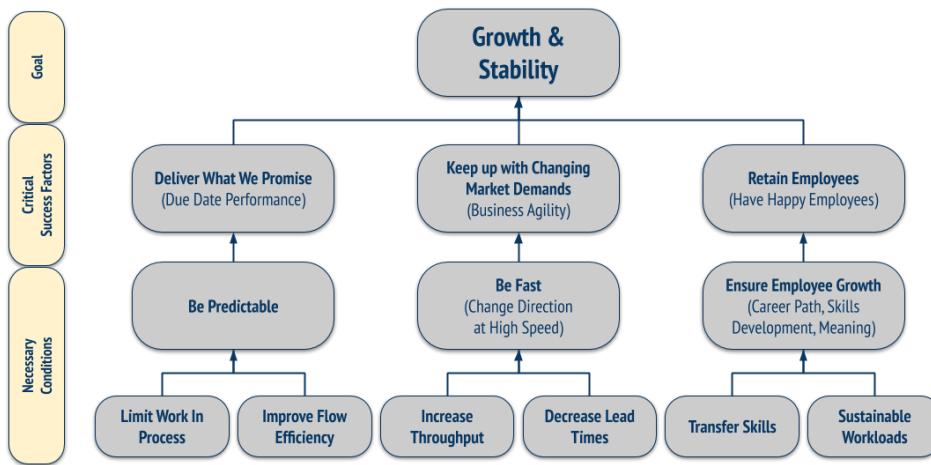
Critical Success factors

What are *some* of the factors that must be present in order to achieve *Growth and Stability*? While we can elaborate and find our own, typically we might say that some *Critical Success Factors* are:

- Deliver what we promise
- Keep up with changing market demands
- Retain employees

Necessary Conditions

We can further decompose the *Critical Success Factors* into any number of *Necessary Conditions*



We can see that many things come together. Here are a few of the key statements that we can read off the *Goal Tree*:

- “In order to achieve growth and stability we must deliver what we promise, keep up with changing market demands, and retain employees.”
- “In order to deliver what we promise, we must be predictable.”
- “In order to be predictable we must limit Work In Process and improve Flow Efficiency.”
- “In order to keep up with changing market demands, we must be fast (so we can change direction at high speed).”
- “In order to be fast, we must increase Throughput and decrease Order to Cash Lead Times.”

And so on.

We leave the articulation of the last branch (“In order to retain employees...”) as an exercise.

It is clear that many of the things we look at when concerned with *Operational Flow* are *Necessary Conditions* supporting the *Critical Success Factors* for achieving the organization’s *Goal*:

- Limit Work In Process
- Improve Flow Efficiency
- Increase Operational Throughput and Financial Throughput
- Decrease Order to Cash Lead Times

All this together makes focusing on the *Goal Tree* so synergistic: the outcome is greater than the sum of the parts.

Assignment

Run a *Goal Setting* workshop for our organization, involving the CEO and all intermediary managers. Create a *Goal Tree* and identify any *Critical Success Factors* and subordinate *Necessary Conditions*.

If we feel that the sample *Goal Tree* shown above is capturing our organization's primary concerns, we may start with it, but we should not limit ourselves to *this tree*. We should develop our own *Goal Tree*, with the active participation of *all* people involved - from the CEO to the latest junior hire.

Remember: leave no one behind.

Expected Effect

A shared *Goal Tree* with the buy-in and commitment of all participants, including our CEO and all intermediary managers.

Pattern 4 - Prime the Mindset

As we have discovered throughout this book, the thinking patterns and habits - the *Mental Models* - that become the accepted norm of our organization are what ultimately sets the limits of its performance.

Some *Mental Models* might be counterintuitive and need deliberate effort to be appreciated, let alone accepted.

From the very outset, it must become clear to *everyone* that the intent is to develop a shared, deep understanding of how things work or could work. One way to ensure we start off with the right attitude and mindset is to establish a frame of reference through these easy to understand metaphors:



The Story of Herbie: A simple story like Herbie's hike can serve as a *Mental Model* about how to think in any situation. If we accept that the entire troop will have to arrive at base camp with no one left behind, we are instilling the idea of *Unity of Purpose*. Likewise, when we are relieving Herbie of his weight and Herbie trusts his peers, we observe the value of a *Community of Trust*.

There has to be a Herbie. It is materially unavoidable: someone (or something) has to be the Herbie of the situation. It is important to understand that there is no dishonor in being Herbie. On the contrary, it gives the whole organization a leverage point on where to focus efforts.

No one is immune to being Herbie. It has to be accepted without any drama, blame or undue pressure. But culture wise, *not* working at full speed all the time is contrary to the expectations of *Cost Accounting* ("*We pay you a salary here, so you better be productive!*").

Both managers and workers need to accept the idea that idle people waiting for work is a *Necessary Condition* for high performing teams. It is when everybody is overloaded that we have queues of work waiting for non constrained resources and the organization as a whole is underperforming.



The Patient in the Hospital: We have all had the experience (either in person or through some close relative or friend) of what it means to be a patient in a hospital. We look at the *Work Load* that is hitting our organization as if it were a patient going to a hospital. We must not think of ourselves as the doctors, nurses and wardens. We must have the perspective of a patient and do everything possible so that they don't have to *wait* for us.

The metaphor of the patient in the surgery theatre is powerful. Once the patient is anaesthetised (i.e. work is started), the entire team will do all in their power to dismiss the patient (i.e. work is ended) as soon as possible.

While the operation is in progress, no distractions are allowed.

This is the kind of attitude that the *entire* organization should have with respect to work that is started. It must be finished as soon as possible, with no interruptions, no multitasking, no distractions. And no compromises or shortcuts on quality.



The Jeep, the Jungle and the Journey: If we set out for a *Journey* across a *Jungle* in a *Jeep*, we will care about having our vehicle in the best mechanical conditions possible, with all fuel, oil and water tanks topped up, and with the right tyre pressure. Everything needs to be top notch. We don't set up for a *Journey* full of surprises in a *Jungle* full of dangers with a *Jeep* that we know might have failures. It just makes sense, doesn't it? The *Jeep* is our *Work Process* which is in our *Span of Control*, and which we need to be in total mastery of. The *Jungle* is the *Work Flow* that has the shape and form of the *Work Load*. The *Journey* is how we actually accomplish our movement in the *Jeep* while travelling through the *Jungle*: it is our *Work Execution*.

In today's challenging and ever changing business environments, it becomes difficult to find guidance and workable management methods. Many surrender by lamenting about the ephemeral control one can exercise over "Complex Adaptive Systems" and resort to sense-making frameworks like Cynefin. These are all valuable ways of thinking, but their application is often misunderstood and misdirected.

Note: One of the most impairing anti-patterns of Agile approaches is the belief that because the domain is unpredictable, then the ways of working must have enough degrees of freedom to become a daily exercise of arbitraging optionalities - often on the basis of intuition and gut feeling, rather than data and knowledge.

The simple metaphor of *The Jeep, the Jungle and the Journey* can bring clarity on how to think about all of this, because it is what allows us to move through a VUCA world all the while adopting high performance working practices.

There must be a clear distinction between the *domain* and the *way of working*. The *domain* - which can be completely unpredictable - and the *way of working* - which can be instituted deliberately through the protocols, ideas, habits that are shaped by the *Mental Models* that we have or adopt.

The *domain* is like the *Jungle* - full of traps and dangers.

The *way of working* is like the *Jeep* - it must be in the best conditions possible. We have the power of establishing what those conditions are (by adopting the *TameFlow Mental Models* and exercise decision-making). The way we decide to work is *entirely under our span of control*, notwithstanding how much VUCA we stand to face.

The *Work Execution* is like the *Journey* that we move along through the *Jungle* by means of our *Jeep*.

With this metaphor, we are setting up the organization to move through the challenges of the world, equipped to use the OODA loop (though that is a topic for another book!).

Assignment

Make sure that everyone involved - from the CEO to the latest junior hire - fully understands *the story of Herbie*, the metaphor of *the patient in the hospital* going to the surgery theatre, and the analogy with the *Jeep*, the *Jungle* and the *Journey*.

Expected Effect

It is accepted that there will be a Herbie - and everyone is prepared, ready and eager to help. Everyone should be ready to help, likewise nobody should be ashamed of being helped or asking for help.

Idle workers are not frowned upon. They are understood as necessary for reaching high performance - those who are faster will wait for those who are slower.

Once work is started, then it is everybody's responsibility to finish it as soon as possible. *Multitasking* and interruptions are to be avoided like the plague.

The way we decide to work should be orthogonal to the domain we are working in. We want to have full control over how we work, and to be able to decide how we work on the basis of a deep understanding shared across all levels of the organization.

Pattern 5 - Measure The Ends and Have a Plot

In the above *Goal Tree*, and based on the *Mental Models* we have discovered throughout this book, we can see that we need to limit *Work in Process*; improve *Flow Efficiency*; increase *Throughput*; and decrease *Order to Cash Lead Times*. But we also need to have *Metrics* that tell us if we are effectively attaining these *Necessary Conditions*.

In most places the *Metrics* will be absent. In this case, the easiest way to kickoff is to measure how long work takes from start to finish. Naturally, this also implies that we have very well defined entry and exit points, so that we know when work has effectively started and ended.

Assignment

Explicitly define the start and end of the *Flow Time* measurements. That is, simply measure the duration of work from start to finish.

Explain to all interested stakeholders why *Flow Time* make sense from a business perspective (as explained in the first few chapters of this book), anticipating the arguments about *Flow Efficiency* and *Constraints Management*.

Learn how to create and interpret *Flow Time Distribution* charts of our own *Work Flow*.

Expected Effect

Our team's *Flow Time* data is presented as a *Flow Time Distribution* chart. The CEO and any interested manager is shown our team's *Flow Time Distribution* chart and understands how to interpret it. The CEO agrees to use the *Flow Time Distribution* charts to establish baselines for individual teams (but certainly not for comparing performance across teams!).

The chart can also be used to identify different kinds of works (or “*Classes of Service*” as they might be called), which can be useful in certain situations. Outliers on the charts will hint at possible *Special Cause Variation*; or - if they repeat - suggest that there might be recurring *Common Cause Variation*.

Note: Other *Metrics* - such as *Flow Efficiency* and *Throughput Rate* - are more appropriate to measure performance across teams or divisions. Knowing to which division we should farm out work to is sound business. Knowing where *Herbie* stands becomes paramount to operating according to *Constraint's Management*. However, when getting started, plotting *Flow Time Distribution* charts is one of the easiest ways to begin using the ideas of *Flow*, and requires no further instrumentations, measurements and calculations (as the other more significant *Metrics* do). As we progress in our journey toward *Flow* we will get to those other *Metrics* too.

Pattern 6 - Show the Work

If the work is not already made visible through a *Kanban Board*, now is a good time to start.

Assignment

Set up a *Kanban Board*, and place it in the work environment (usually on a wall). If we are using any kind of electronic tool, consider using a large screen. However, physical boards typically promote richer interactions between the team members.

Expected Effects

Have a *Kanban Board* in place for the team. The CEO and other interested managers understand the benefits of visualization of work, and they may inspect the *Kanban Board* at any time. The CEO approves of our team's use of the *Kanban Board*.

Pattern 7 - Show the Flow

The *Kanban Board* is good to start with, but if more significant *Flow Metrics* are to be collected - especially in order to support *Work Item Ageing Signals*, *Buffer Signals* and *Flow Efficiency* - then more sophisticated boards become necessary.

Assignment

Transform our boards at least into *Flow Efficiency Boards*, and possibly into *DBR Boards* or *TameFlow Boards* as explained in *Chapter 14 - Flow Efficiency, DBR and TameFlow Kanban Boards*.

Expected Effects

The enhanced boards will allow us to properly collect *Wait Time* and *Touch Time Metrics*, and consequently, to identify the *Constraint* in the *Work Process* represented through that board.

Our *Kanban Board* is instrumented so that *Touch Time* and *Wait Time* can be measured.

Flow Efficiency is constantly calculated and monitored.

Pattern 8 - Show the Numbers

Once the board is properly instrumented, we may start collecting *Flow Metrics* and visualize them.

Distinguish between different ways that *Flow* can be measured. Learn how to interpret the different diagrams and charts. Explain *Little's Law* and its assumptions. Illustrate how *Little's Law* affects predictability. Make the parallel with *Throughput Accounting* which espouses the same conclusions as *Little's Law*.

Make sure that we can distinguish between *Touch Time*, and *Wait Time* and measure them appropriately. Understand how *Flow Efficiency* affects the bottom line, and how *Throughput Rates* are additive and can help us manage across teams, divisions, projects, etc., within the same constrained system.

Assignment

Use an appropriate tool to collect the *Metrics* and provide *Flow* visualization in the form of *Cumulative Flow Diagrams*, *Scatter-plots* and other diagrams. In particular, check out the *Flow Efficiency* metric - provided that it is measured properly. Explain in no uncertain terms to the CEO and all managers why focusing on *Flow Efficiency* is beneficial. Also identify *Flow Time* percentiles and handle *Ageing Signals*, and be prepared to relate the *Ageing Signals* to *Special Cause Variation* and *Common Cause Variation*.

Expected Effects

The CEO and managers understand and know how to interpret such *Metrics* and diagrams. They know how to recognize when performance improvement is occurring, and when there are problems and blockades.

The CEO agrees to the usage of *Flow Metrics* and *Flow Visualization* as a means of communicating, making decisions, and reporting about the state of the work.

The CEO and all managers deeply understand the advantages of first improving *Flow Efficiency* by reducing *Wait Time* (as explained in *Chapter 3 - The Business Value of Flow Efficiency*), and then investing in improving the *Work Process/Touch Time* at the *Constraint* only (as explained in *PART 2 - Acting on the Archimedean Lever that Boosts Performance: the Constraint*, and *PART 3 - Making Accounting Make Money*).

Pattern 9 - Limit Work in Process

Limiting *Work in Process* is arguably the single most important action one can take. It will reduce or eliminate multitasking; reduce batch size; make the system stable and predictable; and is generally the stepping stone on top of which the entire *Flow thinking Mental Models* are built.

While it is relatively easy to agree on the positive effects of limiting *Work in Process*, a consequential question is: how do we do it?

The two most popular approaches in use today are Scrum (and many derivatives like Nexus, LeSS, SAFe, Scrum@Scale, etc.) and *Kanban*. Both methods do indeed limit *Work in Process*. Scrum does it by using *Sprint* timeboxes. *Sprints* limit *Work in Process* through the *Sprint Planning* ceremony, whereby only a certain number of *User Stories* are selected, typically limited by the *Velocity* measured in *Story Points* that the team has had during the last iteration. *Kanban* employs *Column WIP Limits*, which have numerous shortcomings as we have seen in Chapters 9, 11, 12, 14 and 20.

So how can we reasonably limit *Work in Process*?

In the *TameFlow Approach* three options are considered:

1. If we are using a simple *Kanban Board*, never release more *Work Items* into the *Work Process* of a team than there are team members. This option is explained in depth in Chapters 22-25 of the *Hyper-Productive Knowledge Work Performance* book.
2. If we have instrumented a *Kanban Board* as a *DBR Board* or as a *TameFlow Board*, we can implement *DBR Scheduling* on the *Constraint in the Work Process*. In the case of a *TameFlow Board*, the *DBR Scheduling* is further refined with a CONWIP limit where the number of *Replenishment Tokens* is no greater than the number of members on the team. This method is described in detail in Chapter 18 of the *Hyper-Productive Knowledge Work Performance* book.
3. For project portfolio management in a *PEST* environment, the *TameFlow Approach* favours *DBR Scheduling* on the *Constraint in the Work Flow* as described in *Chapter 12 - Drum-Buffer-Rope Scheduling*.

In any case, it is of paramount importance to limit *Work in Process*.

Assignment

Ensure that *Work in Process* gets effectively limited, preferably in one of the three ways suggested by the *TameFlow Approach*.

Expected Effects

As *Work in Process* decreases, workers become less burdened - even idle at times. The *Flow Metrics* and the various diagrams improve dramatically. The CEO and interested managers can appreciate how limiting *Work in Process* effectively supports stability and predictability, to help them keep their promises to clients and markets. The CEO and managers will accept the policy to *Postpone Commitment* in order to limit *Work in Process*.

Pattern 10 - Make **MOVEs**

One of the pillars of the *TameFlow Approach* is to be able to minimize action (the “*amount of work not done*” as it is described by one of the Agile Principles) to have the maximum effect. Hence, a key criterion is knowing when *not* to spring into action.

Examples of conventional factors that result in overactivity - or activity at the wrong time - are:

- The *Kanban Method's* use of *Column WIP Limits* which gives too many false positives, and waste focus and efforts on remediation actions that are generally unnecessary.
- The *Daily Standup* meetings of Scrum, which often degrade to mere ceremonies confirming and validating normal state of “business as usual.”
- The *Sprint Planning Meetings* which have the laudable intent of identifying what development of value could be delivered in the next two weeks - but with no real concern about whether or not a customer valued delivery is smaller or larger than two weeks. This often results in unnecessarily interrupting the *Work Flow* (if the valuable delivery takes more than two weeks), or in incurring an unnecessary *Cost of Delay* (if the valuable delivery takes less than two weeks).
- The *Sprint Reviews and Retrospective Meetings*, which have the same ailments as the *Sprint Planning Meetings*.

Many of these shortcomings are directly addressed with the notion of *MOVEs* as described in *Chapter 15 - Outcomes, Values and Efforts in PEST Environments*. Organizing work in sequential *MOVEs* with a well determined *Target-Scope* will allow us to:

- Use actionable *Work Execution Signals* (*Ageing Signals* and *Buffer Signals*) to get leading indicators of oncoming unfavorable variation.
- Adopt the *Management by Exception* discipline to avoid unnecessary meetings since they are called only on demand when there is a signal justifying them. Planning and/or review meetings are held only in between two distinct *MOVEs* and not on a cadence which might call on more fruitless meetings during execution.

It is the *Target-Scope* of a *MOVE* that allows us to manage by means of *Work Execution Signals* with *Buffer Management* and scheduling of planning and review meetings in a way that does not disrupt the ordinary execution of work. Therefore, it is of essence to organize work in terms of *MOVEs*.

Assignment

Organize all incoming *Work Load* in terms of *MOVEs*, where there is a well defined minimal outcome and/or value to realize a *Target-Scope*.

Expected Effects

Any existing partitioning of incoming work (like Sprint, Work Packages, Epics, Stories, etc.) will be replaced by *MOVEs*. If there is no work partitioning in place, then we introduce the practice via *MOVEs*.

The definition of a *MOVE* clearly takes into account the outcome and/or value to be delivered by its *Target-Scope* on the basis of an economic or service motivation.

The economic motivation requires the *active* involvement of the CFO (or some of their staff).

The CEO and all intermediary managers approve of organizing work in terms of *MOVEs* and support the economic motivation.

Pattern 11 - Forecast (or Estimate)

To have both an indication of the duration of a *MOVE* and to be able to construct its *Buffer*, the amount of work and time needed to realize a *MOVE* needs to be forecasted (or estimated).

Assignment

As suggested in *Chapter 13 - Portfolio Prioritization and Selection in PEST Environments*, we need to adopt some probabilistic forecasting approach (as described by Daniel Vacanti's books) or some lightweight and fast estimation technique (like T-Shirt Sizing).

The forecast needs to be expressed as a probability distribution (rather than a precise numeric estimate) as described in *Chapter 16 - Introduction to Execution Management Signals*.

Even if we use estimation techniques, do not settle for precise numeric estimates, but accept only probability distributions. This is necessary in order to determine the size and position of the *MOVE*'s control *Buffer*.

Expected Effects

Every *MOVE* will have a duration probability distribution forecast as well as an articulated expected outcome or business value.

Pattern 12 - Place and Use the *Buffers*

For every *MOVE* duration estimate, we need to determine the placement and size of its control *Buffer*, as described in *Chapter 16 - Introduction to Execution Management Signals*. Once the *Buffer* is defined, then - on a daily basis - the *Buffer Consumption* and the *Buffer Burn Rate* need to be refreshed, and evaluated as a *Work Execution Signal*.

Assignment

Instrument our work settings and/or electronic tools to effectively implement *Buffer Management*.

Expected Effects

When any given *MOVE* is being realized, its *Buffer Burn Rate* can be computed and known at all times.

Pattern 13 - Bake the Whole Cake

Most of the *Assignments* seen so far are applicable to a single team - the team with which we started working.

Yet, the objective is to come to a comprehensive management approach for a full PEST environment.

There are still more *Assignments* to be undertaken, but they apply to the collection of teams as a whole. So before we can go any further, what has been done up till now needs to be replicated across all teams, so that they are all standing on the same baseline.

Assignment

Repeat the above *Assignments* for all teams in the organization.

Expected Effects

All teams will have appropriately instrumented boards to *Visualize Flow* and collect *Flow Metrics*. All managers between the operational teams and executive management will be versed. All work is managed in terms of *MOVEs* and every *MOVE* is instrumented for *Buffer Management*.

Pattern 14 - Set Priorities and Sequences and Get to Full-Kitting

Once all teams are on the same baseline, we can be more ambitious and really start managing the *PEST* environment. We start with orchestration and the problem of deciding the relative importance and the sequence of the incoming *Work Load*.

When we have a collection of *MOVEs*, all of which have been given an economic valuation (remember that the CFO was involved!) and a duration forecast, we can produce a prioritized and sequenced list, as described in *Chapter 13 - Portfolio Prioritization and Selection in PEST Environments*.

If we are in a *PEST* environment, then every *MOVE* needs to be broken down into the work components that will hit the individual teams. The team with the (overall) longest forecasted queue of *Work Load* can be considered as the *Constraint in the Work Flow*.

The suggested prioritization will be the one derived from the *Throughput Rate* calculation for every *MOVE* with respect to the *Constraint in the Work Flow*.

All this needs to become an ongoing exercise as described in *Chapter 18 - Full-Kitting as Ongoing Executive Activity*.

Hence it is also necessary to introduce the activity of *Full-Kitting*.

The nominal calculated prioritization according to each *MOVE*'s *Throughput Rate* can (and should) be overridden by the human judgement of the *Flow Managers* when necessary. If there are conflicts of priorities between different stakeholders, then those divergences and their respective *MOVEs* need to be escalated for ranking resolution by top executive management as described in *Chapter 18 - Full-Kitting as Ongoing Executive Activity*.

Assignment

Implement *Full-Kitting* and the related *Prioritization* and *Sequencing* policies as described in *Chapter 13 - Portfolio Prioritization and Selection in PEST Environments* and in *Chapter 18 - Full-Kitting as Ongoing Executive Activity*.

The start of work on every *MOVE* by every team should be planned with respect to the virtual *Constraint in the Work Flow*, as described in *Chapter 19 - Execution Management in PEST Environments*.

Expected Effects

The release of work into the *Work Flow* is prioritized and sequenced for maximum economic return based on the *Throughput Rate* of the *Constraint* in the *Work Flow*.

Prioritization is done quickly and automatically - and is then subject to common sense and judgment for sequencing by the *Flow Managers*.

In case of conflict of interest between the *Flow Managers*, the issue is escalated for swift ranking resolution by top management.

Pattern 15 - Keep Feeding Herbie

Once all the teams are instrumented and we can determine their *Work Loads* and identify the *Constraint* in the *Work Flow*, we need to limit the *Work in Process* across *all* the teams, according to the *Productive Capacity* of the *Constraint*.

Note well, and this is worth repeating: the *Work Load* of *all* teams is limited according to the *Productive Capacity* of the *Constraint*.

In other words, we need to implement *Drum-Buffer-Rope Scheduling* at the portfolio level, centered around the *Constraint* in the *Work Flow*; and hence the *Constraint* in the *Work Process* of *that* Herbie team.

Assignment

Implement a *Drum-Buffer-Rope Portfolio Kanban Board* as described in *Chapter 12 - Drum-Buffer-Rope Scheduling* and in *Chapter 18 - Full-Kitting as Ongoing Executive Activity*.

Expected Effects

The number of concurrent projects being worked on will decrease dramatically; each project's *Flow Time* will decrease; and the system's *Throughput* will increase.

Pattern 16 - Catch the Signals

To be quick to react when unfavorable variability is affecting *Work Execution*, the new organizational nervous system must have the necessary trigger receptors in place. Then, everything that is instrumented will be able to generate *Work Execution Signals*:

- The *Work Items* on any single team's *Flow Board* will generate *Ageing Signals* if they are being held back.
- The *MOVE* being worked on by any single team will generate *Buffer Signals* when challenging conditions are met.
- In the aggregate of *Portfolio Management*, the collection of all *MOVES' Buffer Statuses* will highlight where management attention is warranted.

What matters here is understanding the significance of creating *actionable Work Execution Signals* through *Work Visualization* and *Flow Metrics*.

The purpose is to improve overall *Informational Flow* to support decision-making at all levels of the organization.

Assignment

Make use of all avenues to visualize the state of *Work Execution*:

- Visualize the progress of a *MOVE* with *Cumulative Flow Diagrams supported by the relevant Buffer Control Charts*.
- Put in place all necessary Buffers - caring for *Buffer Placement*, *Buffer Sizing* and *Buffer Management*.
- Calculate the *Buffer Consumption* and the *Buffer Burn Rate* for all *Buffers*. The calculation should be automated and available on-demand at all times.
- Detect and observe the *Buffer Signals*.
- Visualize the state of the *Buffers* with *Buffer Fever Charts* and *Buffer Control Charts*.

Make sure to explain to the CEO and all intermediary managers how *Work Execution Signals* work.

Agree with them on escalation policies that might be triggered by the *Work Execution Signals*; how they should be interpreted; and how management should respond.

Expected Effects

Buffers and Buffer Management are in use. Top management are capable to support “*Inspect and Adapt*” and “*Sense and Respond*” with *Buffer Signals* and *Buffer Management* respectively.

The *Work Execution* is monitored in real time and attention is given to detecting *Work Execution Signals*.

The C-level executives are responsive to *Management by Exception* escalations generated by the signals.

The CEO and managers accept and approve of extended escalation policies, and commit to being responsive accordingly.

The organization is no longer captivated by variances from the plan but rather by the state of the *Work Execution* of the *Work Load* that has been released into the *Work Flow*.

Pattern 17 - Stand Up for a Cause

A conventional *Agile/Scrum* practice is to have a daily *Standup* meeting. In *TameFlow*, meetings should be held *when necessary*. Avoiding meetings like *Standup* meetings or *Sprint Planning* meetings makes both economic and logic sense, simply because it prevents wasting workers’ valuable time.

For example, if an organization has 300 engineers, and they all spend 15 minutes a day in *Standup* meetings alone, that translates into 1,500 hours a month of *Capacity* lost.

Note: In this example, let’s not forget the lessons of *Throughput Accounting*. The economic loss is incurred only by those engineers (a minority) that are the current *Constraint* of the system. However, it just does not make sense to use all that excess engineering *Capacity* to hold pointless meetings. Those

1,500 hours a month or so could better be used to help the Herbie of the situation; or to “sharpen the saw.” Just because it costs us nothing to employ *Excess Capacity*, it does not mean we should use it mindlessly.

The hygiene argument is that it simply does not make sense to have such meetings if everything is going well. Similar arguments can be made for the *Sprint* planning meetings, and almost any other meeting that is held in conventional setting. That is the power of *Management by Exception*.

Note: The objection that without such meetings, and especially the daily *Standup*, there will be no opportunities for the team members to socialize and develop their emotional skills is addressed in two ways in *TameFlow*. First, is the awareness that all the *Non-Constraint* teams must accept to be idle if they truly want to *Subordinate* to the *Constraint*. One acceptable way to use such idle time is with socializing activities. (By the same token, we would totally be infuriated to have the *Constraint* team attend any meeting that is not directly relevant to their work execution - they are the *only* ones that cannot afford to waste time in anything but “getting the work done.”) The second is that in order to create more humane working conditions, the organization should actually institutionalize the Swedish ritual of the *Fika* breaks. It is a coffee break (typically in mid-morning and mid-afternoon) that has the specific purpose of socializing, where it is acceptable to speak just about anything. This also brings clarity to the purpose of the actual working meetings: they happen to move the *MOVEs* forward! There is a focused intent. During those meetings team members should have a razor sharp focus on the issues being raised. Chitchat is not expected to take place during *TameFlow* work meetings - but it is mandatory during the *Fika* breaks!

The question then becomes what should trigger meetings.

The *TameFlow* approach will use *Work Execution Signals* to establish if and when a meeting should be called. While there might be a preset time scheduled every day for such meetings, the meeting will be called and held only if warranted.

Typically, there will be two signals to look for:

1. *Work Item Ageing Signals*, as explained at the end of *Chapter 20 - Governance in PEST Environments*.
2. *Buffer Signals*, as explained in *Chapter 16 - Introduction to Execution Management Signals*, *Chapter 19 - Execution Management in PEST Environments*, and *Chapter 20 - Governance in PEST Environments*.

The exception to this policy is when someone discovers new information that has an impact (positive or negative) on the *Work Execution*, even if at that time when there are no signals to worry about. It is in those instances that meetings are called for. For example, if suppliers announce that they will change their way of working. Likewise, if there are defects or bugs reported from the field. Note that any such information might require to change the *Target-Scope* of the *MOVE* that is in process, or some other *MOVE* that is in the pipeline.

Meetings that are held for planning or review purposes should also be called only when needed, and that is specifically from one *MOVE* to the next.

Assignment

If we are using daily *Standup* meetings change the call policy for the meeting so that it is held only if there are relevant *Work Execution Signals*, or other specific reasons that any team member has become aware of.

Also change the policy so that such meetings are *not* timeboxed, as is typically suggested in Agile practices; but rather they go on until the reason for the signals has been uncovered and proper countermeasure or escalation decisions have been taken.

Agree with the CEO (and managers) about the escalation policies based on *Work Execution Signals* and how they are expected to act themselves.

Expected Effects

Less time is spent in meetings, and when meetings are held, they focus on actual problems. Meetings are squarely focused on resolving problems, and not on reporting on status.

Escalation policies are agreed upon and instituted. The CEO and all affected managers understand and approve of the escalation policies, and commit to being responsive. Typically any issues requiring management attention will be brought to their consideration at the weekly *Full-Kit Ranking* meetings; unless the issues are so urgent that they require immediate management insight.

Pattern 18 - Swarm the Block

While *Wait Time* and *Touch Time* are natural components of the *Work Process*, there are moments where work might get stuck in its tracks because of all sorts of problems. With the surgery metaphor in mind, whenever there is a blockade or an impediment preventing work from progressing, in the extreme case the *entire organization* should spring to action to remove the blockage or impediment.

As mentioned above, this should happen based on signals from the *Work Execution*:

- When there are *Ageing Signals* for a *Work Item*, the entire team should swarm onto the problem.
- When there are *Buffer Signals* for the delivery of a *MOVE* by a single team, the team lead and anyone else responsible for that *MOVE* (including other teams) should swarm onto the problem.
- When there are *Buffer Signals* at the *Portfolio* level, then the entire organization should swarm onto the problem.

The notion is that whenever there is a problem, the smallest organizational unit that has the solution to that problem within its *Span of Control* should immediately react.

If the unit does not have control of the solution, it should escalate to its own *Sphere of Influence*, and ask for help.

The ultimate level of escalation will involve the C-level executives and the CEO.

If even they do not have a solution within their *Span of Control*, they might need to act through their *Sphere of Influence* too (which could include suppliers, regulatory institutions, and the market at large).

Assignment

Whenever a blockage or impediment is detected in a *Work Execution Signal*, ensure that the problem is not ignored, and that it is acted upon with alacrity.

Avoid *Flow Backs* and direct any upstream talent to come downstream, if need be - as described in *Chapter 14 - Flow Efficiency, DBR and TameFlow Kanban Boards* and in *Chapter 17 - Introduction to Full-Kitting*.

Expected Effects

Outliers on *Flow Time Distribution* charts will become rarer, and in general *Flow Efficiency* and *Flow Times* will improve.

Pattern 19 - Bubble Baths

Once all teams are effectively using their own *Buffer Management*, we can consolidate all of that information and implement the *Execution Management* and *Governance* in the entire PEST environment, as described in *Chapter 18 - Full-Kitting as Ongoing Executive Activity*, *Chapter 19 - Execution Management in PEST Environments* and *Chapter 20 - Governance in PEST Environments*.

Bubble Fever Charts are the cornerstone to *Management by Exception* for top-level management. In particular *animated Bubble Fever Charts*. These charts will typically be used in steering meetings by whomever is responsible for running the organization's *Portfolio* of projects, initiatives, *MOVEs*, etc.

All managers, department heads and team leads must agree with the gist of Herbie's story: if there is some team that stands out as challenged on the *Bubble Fever Chart*, then it should be helped (as much as is materially possible) by other teams that are in more favorable zones of the chart.

The charts will clearly highlight where the *Constraint* in the *Work Execution* is.

Silo mentality and territoriality need to be eradicated.

The benefits of *Throughput Accounting* need to be understood. The KPIs and middle-manager and team lead reward systems might also need to be revised in favor of *Flow Metrics*.

Assignment

Ensure that all managers across the organization are familiar with *Bubble Fever Charts* and fully understand not only how to interpret them but also how to act on the signals.

Expected Effect

Coordination and synchronization of the multiple projects across the multiple teams will improve dramatically.

The overall *Financial Throughput* of the organization will also increase noticeably.

There will be less infighting and more collaboration.

The team that stands out as the *Constraint* in the *Work Execution* should immediately activate their own *DBR Board*, and try to manage *their* own constraint in the *Work Process* as described in *Chapter 14 - Flow Efficiency, DBR and TameFlow Kanban Boards*.

The *Full-Kitting* activities will also become more sophisticated, and one must become alert to the possibility that the *Constraint* in the *Work Execution* might not be a temporary condition, but that it could become the next overarching *Constraint* in the *Work Flow*, and hence prepare for that situation. The *Throughput* of the detected *Constraint* in the *Work Execution* should be monitored and compared to the *Throughput* of the *Constraint* in the *Work Flow*. If it degrades to the point that it becomes smaller, then it is likely that it is becoming the new *Constraint* in the *Work Flow*.

Pattern 20 - Keep Reason Logs

When we are dealing with knowledge-work, there are huge gains to be reaped if we are able to detect and act on *Common Cause Variation*. However, it is difficult to detect *Common Cause Variation* because in knowledge-work there is so much noise that the signals easily disappear. Or they might be present at random times, or with long intervals between them.

Common Cause Variation is deliberately ignored in knowledge-work. Yet, it is here that the most powerful improvements are to be found.

The institution and maintenance of a *Reason Log* will support long term improvements by addressing systemic *Common Cause Variation*.

Assignment

Whenever there is a relevant *Work Execution Signal* (like *Work Item Ageing Signals* or *Buffer Signals*), make sure to record an entry in a *Reason Log*, as described in the *Hyper-Productive Knowledge Work Performance* book.

A *Reason Log* can be as simple as a spreadsheet with four columns:

- The *Reason* or *Event* that likely caused the *Work Execution Signal* to occur.
- The *Action(s)* that were undertaken to address the situation.
- An updated *Frequency Analysis* of every particular *Reason* or *Event*.
- Whether or not the *Reason* or *Event* is in our *Span on Control* or within our *Sphere of Influence*.

Expected Effects

The team acquires the *habit* of diligently compiling the *Reason Log* any time that a *Work Execution Signal* has caused them to swarm to address some issue.

Pattern 21 - Look Back at the Roots

Whenever a *MOVE* is delivered, it is advisable to conduct a retrospective (or reflective analysis) about how the *MOVE* was executed. The *Agile* literature will describe infinite ways - called practices - in which such retrospectives can be performed. Unfortunately (as with most of *Agile* in general) most of these approaches lack focus.

If a *Reason Log* has been compiled methodically, then it can be used as a valuable source for directing the retrospective.

By performing a *Frequency Analysis* which highlights the most commonly recurring events that affect execution performance, we have a very good clue about where we should look for *Common Cause Variation*.

Notably, when *Frequency Analysis* reveals that there is a particular *Reason* that is recurring, and despite the fact that multiple *Actions* taken at different times did not stop the *Reason* from happening again, then we have a good indication of *Common Cause Variation*. This is indicative that the *Reasons* or *Events* that have been logged are most likely not the real *Root Cause* of the *Common Cause Variation*, because the *Actions* did not stop the recurrences.

To find the *Root Cause* we need to perform a *Root Cause Analysis*.

One of the most effective ways of doing this is with the techniques described in the *Hyper-Productive Knowledge Work Performance* book, employing the *Thinking Processes* of the *Theory of Constraints*.

By using the *Thinking Processes*, we will construct a *Cause-Effect Tree* which will explain why the *Reason* occurred. We will also gain clarity about which elements fall within our *Span of Control* and which are in our *Sphere of Influence*.

If we have to interact with other people to address root causes outside of our *Span of Control*, we will have to master how to manage the *Levels of Disagreement* to improve influence, and to get them to act in our favor.

Assignment

Schedule retrospective reflection events at the end of each *MOVE*, and in particular ensure that:

1. We perform a *Frequency Analysis* (or a *Pareto Analysis*) on the *Reason Log*.
2. We use the outcome of that analysis to direct our *Root Cause Analysis* using the *Thinking Processes* of the *Theory of Constraints* to build *Cause-Effect Trees*.

Inform our CEO (and all managers) about how this process will highlight where and when their intervention is required, especially if action is needed outside our own *Span of Control*, where maybe they might have more immediate capacity to intervene and act directly.

Expected Effects

Root Causes are identified and removed.

Escalations will occur, and top management will have agreed to respond swiftly, whenever such causes are outside our *Span of Control*. In such instances, concerted actions will be agreed upon with the CEO, managers, and other stakeholders.

Pattern 22 - Become a Focused Company

In certain industries, firms cannot escape relying on a single key metric. Dropped call statistics are great for the wireless network operators; and on time departure rate is used across the spectrum in the airline industry. Getting *Metrics* of this quality involves paying for what economists refer to as the “*cost of perfect information*” - and in a cost obsessed company, that will be avoided.

But how do we compete when this one metric is eluding us in our sphere of economic activity? And what about this cost of perfect information? These are two paralysing questions.

This is where we need to stress the inherent simplicity of Dr. Goldratt's teachings. *Throughput Rates* (Financial or Operational) are much less costly to obtain and maintain - and more appropriate for actionable and operational decision support.

Let us master this one metric of *Throughput Rate*; and put it at the top of the pyramid, supported by *Flow Time*, *Flow Efficiency* and *Constraint Management*. We want to maintain this single metric across the organization so that we can always *focus* on what matters, holistically, across the entire organization.

On the path to improvement, we will be reassured that if our *Throughput Rates* increase from period to period, then we are on track to a more profitable future.

Assignment

Ensure we can *effectively measure* the *Throughput Rate* throughout the systems of the entire organization.

Expected Effects

The quality of decision-making using *Throughput Rates* will be easier and quicker.

The use of *Throughput Rates* will outperform *ROI*-based (or other conventional economic calculations) and *Cost of Delay* choices immediately.

Takeaways

As mentioned at the beginning of this chapter, the above patterns must not be considered as a recipe or a cookie cutter. They should just inspire our thinking and prompt us into action - but always under the mindful awareness about which *Mental Models* are at work in our decision-making processes.

We will have arrived at a state of *Focused Flow* if we observe the following:

- All work is performed according to prioritization established or agreed upon during the *Full-Kitting* activity.
- Work is not subject to undue interruptions.
- Issues are detected early, escalated and responded to rapidly.
- Decisions are made cooperatively and with unanimity because there is awareness about and focus on the *Constraint*.
- *Multitasking* is reduced.
- *Work Load* is sustainable for everyone, and the work “flows” naturally throughout the organization.
- Management attention is focused and effective.
- Meetings are called on a *Management by Exception* basis.
- Performance improves significantly.
- Knowledge-workers are either working or waiting for work. Both states being totally acceptable.
- Workers are moved as needed to support the *Constraint*, when and if it is possible.

This list is not exhaustive. There are many more positive effects to be observed in an organization that “flows.” Yet they give us an idea of what it might be. If we pursue the initial patterns listed in this chapter, we will rapidly see our organization move into a *state of flow*.

Stuart Burchill says...

Over the last 20 years, most of us have participated in one way or another towards the beneficial mutation of organizations via the application of principles and practices associated with scrum, XP, scaled agile, kanban, and more recently business agility. At each evolutionary stage, it was deemed necessary to apply the required 'medicine' to cure the ailing organization.

The 'medicine' that worked wonders in one place was not always as effective elsewhere, all the while requiring consumption of specific regimes of foreign concepts, labels, and roles. Were these evolutionary stages of disruptive agile therapy necessary? Back then, the answer was likely "yes" for most organizations that were brittle and caustic to any change.

The prescription was typically ingested with the desperate hope of a cure, rather than a guarantee. Essentially a wishful and expensive IOU attached to a hard-to-swallow pill.

Thereafter, if the cure wasn't forthcoming, the patient was too often blamed for lack of adherence to the prescribed therapy rather than the therapy itself being questioned.

20 years on, and we are still too often applying the same broad-brush medicines, hoping for a universal cure...

EUREKA!

Along comes TameFlow - an evidence-based, light-touch, laser-precision approach able to both diagnose and cure organizational illnesses.

This light-weight sniper approach to finding constraints and eradicating them can be applied at every level of the organization, and can result in a win literally within hours of targeted application.

Finally, a step-function evolution has arrived to deliver on the promise of swift results for minimum effort and disruption.

I applaud Steve and Daniel for their perseverance to bring this refined approach to those of us who have been searching for the right tools. The beauty is that these tools are also timeless, and will therefore retain their razor-sharp edge indefinitely.

Stuart Burchill, Enterprise Strategy & Transformation Consultant / Executive Advisory Services / Lean Agile Coach & Trainer, CTT, SAFe-SPC, CSP, CSM, PMP, BEng

Epilogue - It is Never "Done!"

The last column on a *Kanban Board* is often called the "*Done*" column.

We are now at the end of this book.

We have already put in a lot of effort if we have read this far. A journey always brings new discoveries, and at the end, a certain sense of accomplishment.

We might think: "*At last, we are "Done!"*"

But is this the end? And are we "Done?"

What Have we "Done?"

If we strive for *Continuous Improvement* we know that we are never "Done" - as Taiichi Ohno and Dr. Goldratt taught us. There will always be room for betterment.

Likewise, the *TameFlow Approach* is a living body of knowledge that evolves and adapts continuously, at times, discarding ideas that are past their time, and at other times trying out new ways. However, in all its liveliness, the *TameFlow Approach* has some fundamental pillars or overarching perspectives - actually *Patterns* - which are applicable in most cases.

There are many ways to tame the complexity of *PEST* environments. Agile approaches seem to fight complexity by adding complexity, and by not adhering to modern management science. Traditional Agile approaches assume that the human factor and social interactions will be sufficient to lead to desired outcomes. As traditional Agile claims, every step taken is - by definition - on a path never travelled before.

Modern management knowledge speaks last. This is where the legacy of Dr. Goldratt comes into play. By legacy, we mean the inherent simplicity of both the *Theory of Constraints* and *Throughput Accounting*.

There is no denying that market demand changes without any advance warning.

And our *Capacity* to deliver is a *Constraint*.

What are we to do?

The *Theory of Constraints* thinking that we have applied thus far has to be considered as the obvious solution in *PEST* governance, as our *Capacity* to deliver must be viewed as something malleable that we can adjust dynamically everyday, at practically no cost except that of learning - for the simple reason that unlike the *VUCA* business domains we might be acting in, our *Capacity* to deliver is *entirely* within our *Span of Control*.

The methods discussed in this book should form part of how we all go about our business, as the approach is:

- **Agile:** the *Tameflow Approach* uses a simple definition for agility: the ability to *change direction at high speed*. This both creates and requires *Flow* at the same time. Other agile approaches do not practice and can seldom claim to lead - much less support - agility in its purest sense.
- **Data-driven focus:** Agile coaches come equipped with skills that are necessary. A system's view should always be in the agile coaches' purview, but seldom is. Data-driven management is integral to the *Tameflow Approach*. Yet data-driven management just cannot just fit within most of the agile frameworks that are in fashion nowadays. The human brain has a tendency to overestimate its capabilities; be an easy prey to anchoring; fall into group think; be helpless in integrating data; and get emotional. The antidote to these ailments is metrics.

- **Humane:** Agile purists cringe at the idea of using metrics, and perceive it as a whip to flog the modern times knowledge-worker slaves. What they miss is the realization that metrics give us the insight to receive the gift of identifying the *Constraint* and seeing and managing *Excess Capacity*. In creating the material time for people to focus on something other than *constant firefighting*, the *TameFlow Approach* creates the space for more humane organizations to develop. It is in this space that the practices of *Psychological Flow* can bring human factors to new heights, even beyond what conventional *Agilists* can ever dream to achieve. Why? Because the *TameFlow Approach* addresses the inherent *conflicts* (anchored in the practices of *Cost Accounting*) that all mainstream Agile approaches simply are oblivious to. How can traditional *Agilists* ever hope to make organizations more humane, if they are not able to practically address the most basic and ubiquitous conflicts that are structurally present in almost all organizations due to *Cost Accounting* practices?
- **Management by Exception:** Is there any notion of *Management by Exception* in the widespread practices of common Agile? Taming complexity with complexity forfeits this avenue. Yes, visualisation is really potent and can help us to eliminate "Status Meetings" - but what about all the other meetings, in particular the decision-making meetings? If we have ever been in a portfolio management meeting, 90% of the conversations are repetitions of past or future convocations. The cost of decision-making defeats the purpose of decision-making. It is therefore wise to use all of the *Tameflow Approach's* leading indicators to address overburdening and get the right people - and no one else - at the right place and at the right time with the right information, in order to have the right discussions. This is another way to achieve respect for people, which some Agile implementations have reduced to face value ceremonies with mandatory participation.
- **Leading Indicators:** There is no other source of managerial knowledge, other than the *Tameflow Approach*, that relies on so many leading indicators of risk materialisation for project execution. Agile frameworks come equipped with new roles, artefacts, deliverables and structures. Why don't we all instead learn about *Flow Metrics* and master the use of the *Work Execution Signals* and *Leading Indicators* as proposed in this book? It would cost less and empower all participants in the system.
- **Pattern based:** Tameflow is not a directionless evolutionary approach like the *Kanban Method* or the *Kanban Maturity Model*. It is far more efficient. Why? Because it is structured by a *Pattern Language*. The random walk of evolutionary improvement is replaced by a structured traversal of a *Pattern Network*, and directed toward the *Goal* of the company, while creating *Unity of Purpose* and fostering a *Community of Trust*. If done intensely, the realization of the patterns achieves superior effect than many random evolutionary steps, and in much less time. From the outside, they happen so fast that they might even appear like a "Big Bang" transformation. From the inside, they appear as evolutionary, yet they are directed toward the *Goal*, without the risks and high stakes that are typical of "Big Bang" approaches, nor the random path wanderings of evolutionary change.

Governance is often time bloated with rituals (and a full array of indirect costs). Agile should embrace lightweight avenues to governance where trust can absorb complexity. Hard facts and metrics can build trust and this is where conservative Agile frameworks just makes the assumption that adhering to "principles and values" will be sufficient to reduce friction and achieve desired outcomes. It is just not the case.

The *TameFlow Approach* is an alternative we offer for consideration. While we have been delving into many details of *Operational* and *Financial Throughput* and *Flow*, what happens in the background is not that evident.

The Prevalence of Mental Models

In *The Essence of TameFlow* book, we can read that organizational performance is determined by the organization's ability to *make decisions*. Decisions are not only those made at the C-level executives but encompass also each and everyone's participation at each level of the organization, at all times.

The key is that *all* such decisions should not be contradictory, nor should they produce any sort of conflict between any two individuals or parts of the business. Naturally this requires all decision makers to develop the pattern of *Unity of Purpose*, and that starts by having the same shared and beneficial *Mental Models* which can give everybody a consistent perception of all the problems that are being faced.

If we consider the topics examined in this book - and their impact on employee morale and the bottom line - we can see that from *all* of the things we have learned, it essentially boils down to a *very few decisions*:

- When do we start work?
- How do we prioritize and sequence work?
- How do we select work?
- Where do we focus attention, effort, and resources?
- What is the impact on *Financial Throughput*?

How is it possible that only five simple questions can have such a huge impact? Especially considering the gazillion *Key Performance Indicators* that are regularly monitored with other approaches? What is the difference?

It's the inherent simplicity!

We expect decision-making criteria to be well known throughout the entire organization. That way delegation and autonomy become child's play. Confidence sets in at all levels toward a shared and common *Goal*.

In virtue of the shared *Mental Models*, reality will be understood and interpreted within a common frame, and decisions will happen faster. There will be no debating and there will be no time wasted in seeking consensus because everybody knows what the right decision should be.

Everybody is playing the same game, on the same field, with the same rules.

Speed of execution is critical, but so is the speed and quality of decision-making. When looking at reality, everybody needs to see the same thing. This - in particular - extends to having a shared understanding of the signals that will direct attention to when and where "stuff happens."

We can then ask if it is something special (*Special Cause Variation*) and remain calm and cool, and spring to action only when it will effectively need some counteraction. Differentiating between noise and signals is also a form of respect toward knowledge-workers' time and what they have to pursue. If our system is not equipped to allow for this, we will not be able to progress effortlessly toward our *Goal*, because all and everyone will be interrupted and distracted all the time for every tiniest blip that comes in our way.

Or we can see if this is something that occurs over and over again (*Common Cause Variation*), and then strive to find the critical *Root Cause* that can be eliminated and reap great economic benefits in doing so.

In either case, the organization responds as a whole, as one living organism that reacts to environmental stimuli in one way, and not in a multitude of ways that tear it apart.

Mindsets and Attitudes to Win in a VUCA World

We cannot *do anything about* VUCA (Variability, Uncertainty, Complexity, Ambiguity). The best we can do is to *be prepared* - like *Herbie*, the good Boy Scout! The Cynefin framework gives guidance on the kind of

environment we are dealing with. Some practitioners have proposed the weird idea to actually change approach, methodology and process, as the environment changes. This just does not make sense.

It is evident that existing approaches fit better for certain environments than others. The heavier the approach, the more it moves toward the *Simple* domain. The more empirical the approach, the better it is for the *Complicated* or *Complex* Domain. Almost no approach seems able to fit in the *Chaotic* domain.

The *TameFlow Approach* views the issue in terms of the metaphor of the *Jeep*, the *Jungle* and the *Journey*. We can assume that we will have to confront the most uncertain domain (the *Jungle*). Likewise, we want to be able to confidently rely on our means of surviving in that *Jungle* - which is the *Jeep* (i.e. our entire organization). Our *Jeep* is entirely under our control; it is very well known to us in all of its details; and it handles according to well known performance parameters, with no surprises (unless there's some failure, of course). The *Jeep* is entirely within our *Span of Control*. It is literally in our hands. We can build it so it behaves as we want. Then we have a *Journey* - that is how the travelling through the *Jungle* actually happens by means of the *Jeep*.

The gist of this metaphor is that the *Jeep* corresponds to our *Work Process* which is well attuned and under control; the *Jungle* is the *Work Flow* that comes our way full of uncertainties and dangers; the *Journey* is the *Work Execution* that we perform to move our *Jeep* through the *Jungle* but that can encounter hiccups, glitches, failures, defects and even emergencies.

The development of empirical process control approaches to manage knowledge-work, like *Scrum* and other variants of *Agile*, have the assumption that in knowledge-work predictability is not possible and that one has to opt for adaptability. This is correct in principle, but the proverbial devil is in the details.

The problem is that this conflates the domain (the *Jungle*) and the means (the *Jeep*) of confronting the domain, and also the contingent situations of performing in the domain at a particular time and context (the *Journey*).

The *Jungle* and the *Journey* are unpredictable and to navigate through them, constant adaptability is required. That's where feedback loops play their role in providing the underpinning of empirical process control. And the tighter and more reactive those feedback loops are, the better we can confront all of this uncertainty. Naturally, after reading this book, the *Leading Indicators of TameFlow* will give us the most reactive anticipatory feedback mechanisms in lieu of having to wait for the next review or retrospective ceremony.

While the *Jungle* is an unpredictable environment and the *Journey* brings us unpredictable events, the *Jeep* can and should work according to very well known parameters, with known handling and responsiveness in all situations. We are in control of how we work. We decide "*how we do things around here!*" - we can decide when to start and stop work, and when to adopt new *Mental Models*. We can decide how to prioritize. We can establish the policies of when to call meetings. We can establish the criteria of where and when to focus our attention; when to make decisions; and how to execute them. And, above all, we can learn and have the freedom to improve the "*way we do things around here*" on our own terms - we will always tweak and tune our *Jeep* so that it performs the best *for us* - and not feel obliged to conform to some preset immutable and almost sacred rulebook about how things should be done. (Approaches such as *Scrum* and *SAFe* trigger resistance to change by their "showing" us the "right way" to do things when maybe our company has been in business for longer than *Agile* has been around. Showing respect for how businesses operate is missing in mainstream *Agile*.)

The classic *Agile* world, by conflating the domain, the means, and the performance will bring any management attempt astray bereft as it is of any of the scientific method pillars. Because of these amalgams, *Agile imposes unpredictability even on the part of the system that is stable and predictable*.

Yet, paradoxically, despite the invocation of the "inspect and adapt" mantra, *Agile* approaches - and *Scrum* in particular - are very dogmatic about how they should work. Seldom does it occur that the very process itself needs to be inspected, adapted and *changed*. In these instances, there is no double-loop

learning; but just lip-service conformance to an empty edict. When we say that we are in control of the *Jeep*, we mean not only that we are in control of driving it; but we are also in control of how we *build it*. The *TameFlow Approach* is constantly inspecting and adapting itself, and evolves in a *pragmatic* manner, where theory shapes practice, and practice informs theory.

What needs to be cultivated is the ability to have an organization whose *Capacity* is known and assured in a dynamic way, as it moves through the problems of the domain. Most approaches do not deliver such predictability of performance because they *induce even more uncertainty in how things are done*.

If the system is predictable in its performance, then we can throw it into the most uncertain domain - into the wildest of *Jungles* along the most surprising *Journey* - and yet feel perfectly comfortable and in control that our *Jeep* will respond to our governance as we expect. That's the whole point of what we have been describing in this book - the expectation of reliable and responsive governance - and not the unwarranted wishful thinking of imposing certainty over the uncertain (as conventional project management approaches do). Or to renounce any hope of governance assuming that even our own *Work Processes* of choice are complex and can only be managed via empirical processes control (as Agile does), yet condemned to never change.

The system will be extremely responsive, and we can adapt the course according to the terrain.

We can *change direction at high speed*, which is the essence of *Business Agility*.

The more capable we are of changing direction at high speed, the easier we can adapt to the uncertainty of the domain and the more Agile we become.

The Cynefin model is a high level way of making sense of various domains. In practice, what makes the most sense is to assume that the domain in front of us will be like the most demanding challenges, full of traps, surprises and hostile beasts or enemies.

We can prepare to navigate that terrain just like a jet fighter pilot applies an OODA (Observe, Orient, Decide Act) loop and be confident that our machinery is as responsive and powerful as it can be - just like jet fighter pilots have confidence in their aircrafts.

This is similar to how top sports teams prepare for their competitions. They cannot do anything about how any single game will turn out to be, but they can prepare before they get onto the field. They prepare through training and drills, so that once they are in the heat of the competition, the routines developed in training will kick in due to muscle memory. This is where the notion of *Psychological Flow* - and in particular collective or team *Psychological Flow* - becomes so important.

If a team notices an item with *Ageing Signals* becoming red, there should be an immediate and almost reflexive response, and the issue should be escalated. At any step in the escalation, an equal immediate reflexive response should exist, until the issue reaches a level where it can be addressed. It could go as high as the CEO. But the entire organization is attuned to the idea that *performance is critical for surviving in a VUCA world*, and that being *individually responsive* is part of what makes the whole machinery responsive.

With organizational performance being stable, predictable, and in control, the organization can confront any domain that is unstable, uncertain and even unknown.

Note: The organization can employ any method or methodology it wants, *as long as the shared Mental Models and decision-making criteria are employed to support consistent and rapid decision-making at all levels*. A corollary is that if the organization becomes aware of the fact that their current ways of working are not based on shared *Mental Models* - and especially if the present habits and patterns of thoughts give rise to conflicts within the organization - then the organization should be ready to ditch their existing approaches.

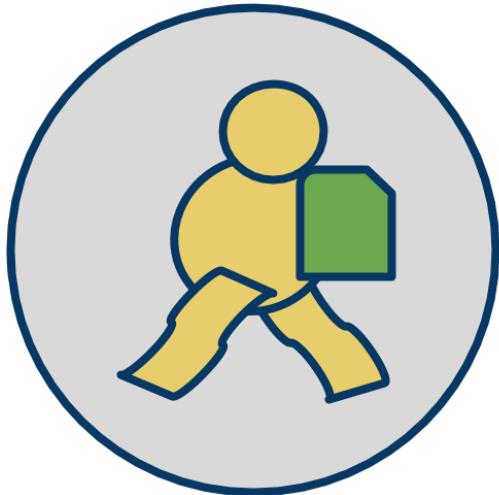
It is clear that the metrics-driven approach will suggest ways to improve, so no matter where the organization stands when adopting the *TameFlow* mindset, its processes will evolve. Such evolution can happen very quickly, even at scale, it all depends on how successful and how swift the company is at adopting the shared *Mental Models*.

This must also be put in relation to the capability and speed at which the organization can *innovate*. Since the world is changing at such a steadfast pace, it is clear that those organizations that can develop solutions and innovate faster will be at an advantage. An organization that works ten times faster than the competition can afford to develop and fail nine times before the others are ready to try their first solution. The opportunities to explore, pivot and adapt are afforded only to those who can perform better than their competitors. Organizational hyper-performance is a critical factor in determining our organization's chances of succeeding in today's fast moving world.

Good luck in your quest for hyper-performance with the *TameFlow Approach*!

... and remember:

***HERBIE IS
YOUR BEST
FRIEND!***



Matt Barcomb says...

Honestly, this book surprised me. Why?

Because I've been reading books about and consulting on Flow, Lean, Systems, Kanban, etc. for over 10 years so I didn't expect to learn much. But how Steve and Daniel blend these concepts and present the interactions between them gave me food for thought on multiple occasions.

That being said, it's also a great read for someone just beginning their exploration into the ocean of these ideas. There's truly something for everyone here.

Matt Barcomb, Founder & Principal Consultant, Intentionally Adaptive - Seasoned Product Leader who enjoys growing lean-agile organizations.

Minton Brooks says...

With "Tame your Work Flow", Tendon and Doiron raise the bar on Agile - the ability to change direction at speed. The Tameflow Approach brings accelerated and dramatic enterprise business improvements. It is not yet broadly used in the industry, but it deserves to be for the sake of making business more humane and the Agile market more sane!

Minton Brooks, Enterprise flow consultant and empiricist Agile coach; credentialed in Kanban Method, XSCALE, Okaloa FlowLab, AgendaShift and the Scaled Agile Framework.

Bibliography

- Anderson, David and Bozheva, Teodora. *Kanban Maturity Model: Evolving fit-for-purpose organizations (Beta Release)*. Lean Kanban University Press, 2018.
- Anderson, David. *Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results. The Coad Series*, Prentice Hall. 2004
- Anderson, David. *Lessons in Agile Management*. Blue Hole Press. 2012
- Corbett, Thomas. *Throughput Accounting*. North River Press. 1998
- Caspari, John A. and Caspari, Pamela. *Management dynamics: merging constraints accounting to drive improvement*. John Wiley & Sons, Inc. 2004
- Csikszentmihalyi, Mihaly. *Flow: The Psychology of Optimal Experience*. HarperCollins Publishers Inc. 2001
- De Grandis, Dominica. *Making Work Visible*. IT Revolution Press. 2017
- Du Plooy, Etienne. *Throughput Accounting Techniques*. General Media Press. 2018
- Goldrat, Eliyahu and Schragenheim Eli. *Necessary but not Sufficient: A Theory of Constraint Business Novel*. North River Press. October 2000
- Fabok, Zsolt. *Impact of Low Flow Efficiency*. Lean Agile Scotland, Sept 2012 & Lean Kanban France, Oct 2012
- Forss, Hakan. *Impact of Low Flow Efficiency*. Lean Kanban France, Oct 2013
- Hodes, David V. *More than just Work*.
- Jaeck, Pierre. *Pratiques managériales et 'Theory of Constraints'*. Thèse de doctorat présentée et soutenue publiquement le 3 juillet 2014
- Katikar, R. *Cost analysis for 'make-or-buy' decisions for manufacturing industries*. International Research Journal of Commerce, Business and Social Science (IRJCBSS) VOL 1, no 9, pp 151-156 (ISSN 2277-9310) Dec 2012
- Klaus, Leopold. *Practical Kanban*. LEANability PRESS 2017
- Land, Dr Lisa and Beau, S. Gana. *Back to TOC: Throughput Accounting*. 2018 TOCICO Webinar. 2018
- Mullainathan, Sendhil and Shafir, Eldar. *Scarcity: The new science of having less and how it defines our lives*. Picador. 2014
- Oswald, Alfred and Muller Wolfram. *Management 4.0 - Handbook for Agile Practices*. Release 3.0. 2019
- Reinertsen, Donald. *The Principles of Product Development Flow: Second Generation Lean Product Development*. Celeritas Publishing. 2009
- Tendon, Steve and Mueller, Wolfram. *Hyper-productive Knowledge Work Performance, The TameFlow Approach and Its Application to Scrum and Kanban*. J. Ross Publishing. 2015.
- Vacanti, Daniel. *Actionable metrics for Predictability*. Amazon Digital Services. 2015
- Vacanti, Daniel. *When will it be done. Lean-Agile Forecasting to answer your customers' most important question*. Actionable Agile Press. 2018
- Weinberg, Gerald M. *Quality Software Management: Systems Thinking*. Dorsett House. 1991
- Weinberg, Gerald M. *Quality Software Management: Congruent Actions*. Dorsett House. 1994
- Woeppel, Mark. *How to grow profit with Throughput Accounting - A Pinnacle Strategies White Paper*. Pinnacle Strategies.
- For a more extensive list of bibliographic references see the TameFlow Bibliography:
<https://tameflow.com/bibliography/>

About the Authors

Steve Tendon



With a background in software engineering (in his early career he lead the development of software applications in diverse fields, like: banking, health care, legal, human resources, and more), Steve is the creator of the [TameFlow ® Approach](#), a systems thinking approach for creating *breakthrough performance innovation* in knowledge-intensive digital businesses. The TameFlow Approach has been developed and used with great success since 2003, across numerous industries. Steve holds MSc in *Software Project Management* with the University of Aberdeen, a *MIT Fintech Innovation: Future Commerce* certificate with the Massachusetts Institute of Technology, and an *Oxford Blockchain Strategy Programme* certificate with the Oxford Saïd Business School.

Daniel Doiron



Daniel has been involved in IT since 1981 in a wide range of roles and responsibilities, primarily in client-facing consulting projects covering government, banking, insurance, and telecom industries to name a few. Daniel's involvement with Agile started with Scrum in 2005 and more recently with Kanban and Management 3.0.

He is heavily involved with Steve Tendon's Tameflow Approach. He is proficient and has working expertise in Finance/Accounting/Managerial control (MBA-CPA-CMA), Agility (CSP), Project Management (PMP), Kanban (CKC and CKP) coupled with 38 years in IT (Bachelor studies & career).

He loves systems and enjoys measuring improvement while embracing teamwork that actually works! Read more on *Tameflow Kanban* and classes availability worldwide at:

<http://agileagonist.com>

Resources

In the following pages you can see some further resources regarding the *TameFlow Approach*.

Community

If you have any questions you would like to ask, or reflections you would like to share about the *TameFlow Approach*, you can join the **TameFlow Community** here:

<https://community.tameflow.com/>

The site gives discussion space not only to the *TameFlow Approach* as such, but also to other related topics, including:

- Kanban
- Lean
- Agile
- Scrum
- Project Management
- Personal Growth
- Tools

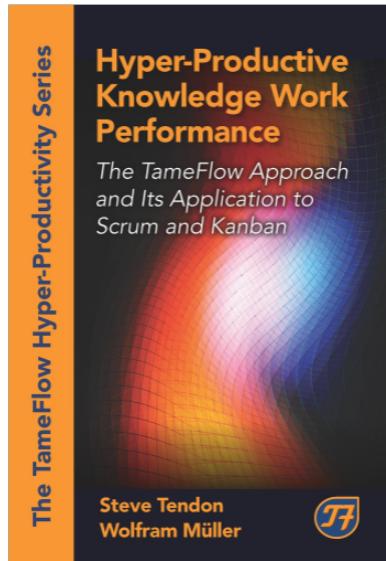
And others too. Join the conversation!

Professional Service, Executive Education and Training

For further information on where to find professional help with the *TameFlow Approach*, explore the pages here:

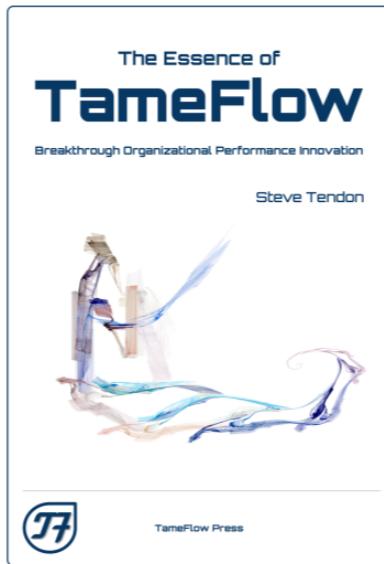
<https://tameflow.com/>

Hyper-Productive Knowledge-Work Performance



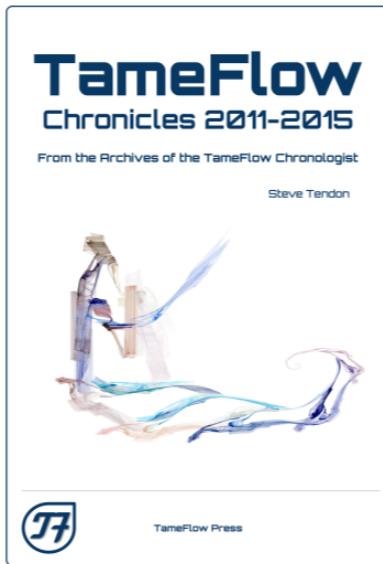
If you want to know more about how the topics covered in this book have been initially developed and what the background ideas are, please consider reading the [Hyper-Productive Knowledge-Work Performance](#) book.

The Essence of TameFlow



While both this book and the *Hyper-Productive Knowledge-Work Performance* book delve into many practical aspects and give you actionable ideas and advice, you will find a more philosophical approach to the founding concepts that underlie TameFlow in [The Essence of TameFlow](#).

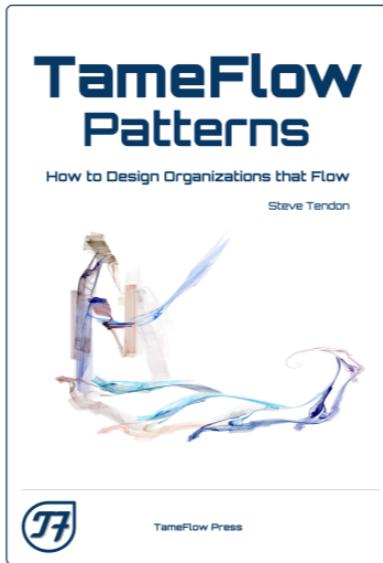
TameFlow Chronicles 2011-2015



The *TameFlow Chronicles 2011-2015* book is a collection of all blog posts published on the “*TameFlow Chronologist*” blog between 2011 and 2015, for ease of reference and convenience of reading. These blog posts were the first public instance of the ideas of the *TameFlow Approach* and eventually evolved into the other books listed here. They can also be found here:

<https://tameflow.com/blog/>

TameFlow Patterns



The [TameFlow Patterns, How to Design Organizations that Flow](#) book outlines how you can describe and design organizations that can reach flow states. The book uses design patterns as the fundamental element for understanding and building such organizations. It covers more of the *Informational Flow* and *Psychological Flow* aspects of the *TameFlow Approach*.

At the time of the publication of the present book, the *TameFlow Patterns* book is still being written.

Help with TameFlow

TameFlow Consulting Limited (<https://tameflow.com>) is the company that supports entrepreneurs and practitioners in adopting the *TameFlow Approach*.

At TameFlow Consulting Limited we provide expertise in the following areas:

- **Process Performance** – We assist businesses to rapidly evolve their current processes to heightened levels of performance by focusing on operational flow improvements.
- **Constraints Management** – We help businesses improve their financial performance by exposing, managing and breaking any latent constraints that limit financial flow.
- **Organizational Performance Design** – We assist businesses to expose and improve their effective structures of communication and interaction for a smoother internal informational flow.
- **Performance Mindset & Attitude Development** – We provide the necessary tools for the staff of the organization to gain an awareness of how their own beliefs and values impact performance and we guide the organization to activate psychological flow.
- **Breakthrough Performance Innovation** – We advise about how to explicitly and deliberately align the four performance flows to produce a real breakthrough innovation in the organizational performance.

If you need help in any matter regarding TameFlow, get in touch with TameFlow Consulting Limited.

Free Bonuses!

Thank you for looking into this book. The book has associated a number of free bonus resources that you may access by visiting:

<https://tameflow.com/tame-your-work-flow-free-bonus/>

You will find:

- Free copy of the book *The TameFlow Games*
- Free copy of the book *TameFlow Chronicles 2011-2015*
- Free copy of the book *TameFlow Patterns*