The Project Management for IT Projects:

AGILE METHODOLOGIES OVERVIEW WITHOUT SCRUM

LESSON 3

Mario Salano

April – May 2025

The Project Management for IT Projects: part 2
INNOVATIVE PROJECTS

The second part of the course is focused on innovative projects.

Steps:

- Basic Concepts: INNOVATION, AGILITY, VALUE DRIVEN DELIVERY
- How to run a Project according to AGILE, LEAN and DESIGN THINKING (the 3 most important methodologies)
- A real life experience

Course agenda (part 2:INNOVATIVE PROJECTS)

1. INNOVATION AND METHODOLOGIES
2. AGILE CONCEPTS
3. **AGILE METHODOLOGIES OVERVIEW WITHOUT SCRUM**
4. SCRUM
5. LEAN
6. DESIGN THINKING
7. VALUE DRIVEN DELIVERY
8. STAKEHOLDERS,TEAMS,ADAPTIVE PLANNING
9. CASE STUDIES
10. EXERCISES
11. CONTINUOUS IMPROVEMENT AND
12. FINAL REVIEW

OBJECTIVES

- **SUPPORT TO STUDENTS** TO FACE THE REAL WORK LIFE OF TODAY
- IT IS NOT ANYMORE ENOUGH GOOD WILL AND PREPARATION
- EFFECTIVE METHODS AND THE AWARENESS OF A LIFE LEARNING ARE NEEDED

THE WORLD IS MORE AND MORE “PROJECTIZED”

SO PROJECT MANAGEMENT, WITH UPDATES, BUT WITH ITS BASIC CULTURE, BECOMES A FORMIDABLE ASSET

PREVIEW OF TODAY **LESSON 3:**
AGILE METHODOLOGIES OVERVIEW (WITHOUT SCRUM)

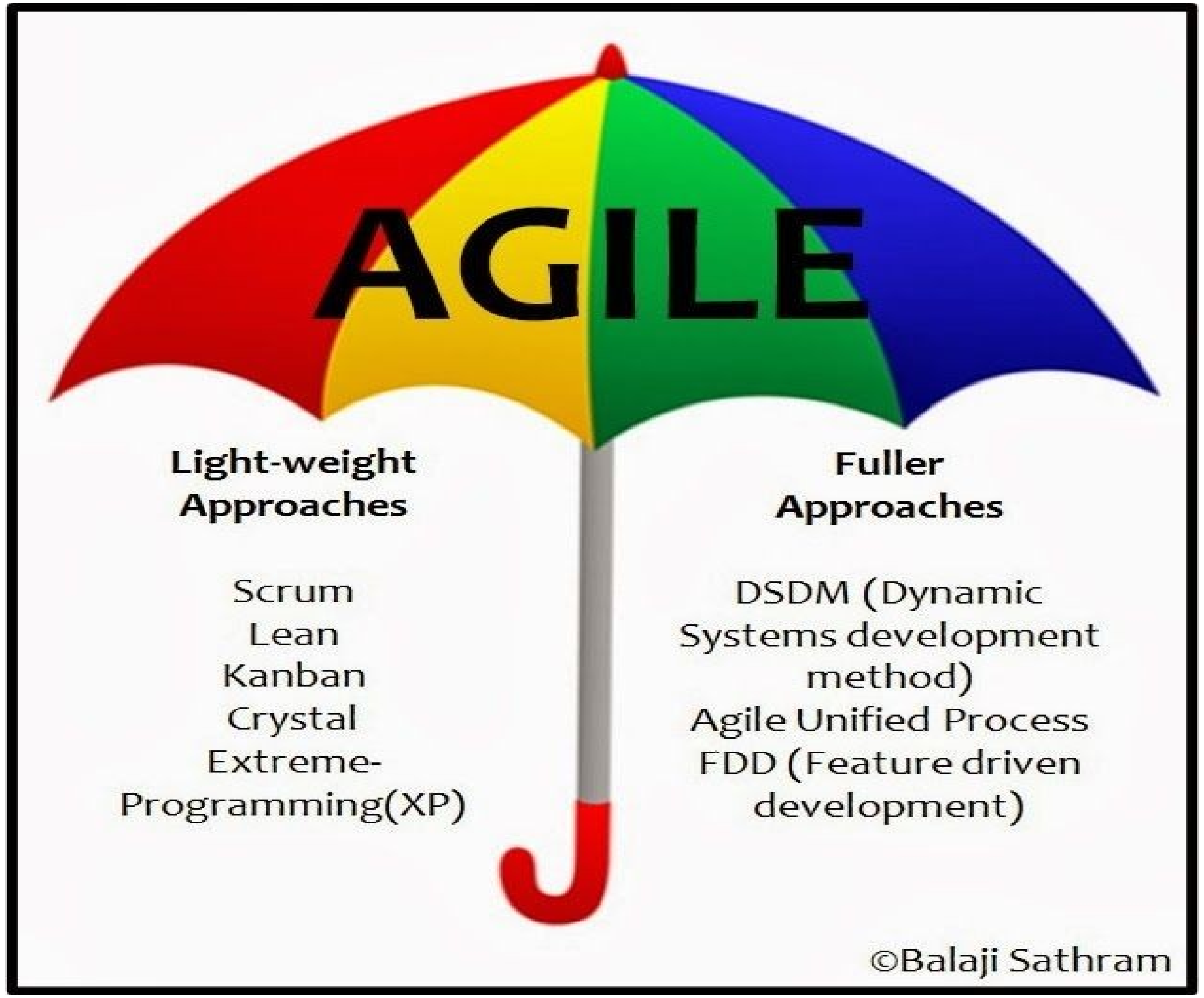
1. WE will go through **AGILE METHODOLOGIES (**without SCRUM as a dedicated lesson will be held on it) but +hybrid
2. Project Management is the main discipline **to do**

AGILE METHODOLOGIES

Different types of projects require different **METHODS**

GOAL:give clients a non stop flow of **VALUE**

AGILE METHODOLOGIES



Light-weight Approaches

Scrum
Lean
Kanban
Crystal
Extreme-Programming(XP)

Fuller Approaches

DSDM (Dynamic Systems development method)
Agile Unified Process
FDD (Feature driven development)

©Balaji Sathram

8+1 MAIN METHODOLOGIES

SOME SPECIFIC TERMINOLOGY

- SPIKE: A SHORT TIME INTERVAL ,DURING WHICH A TEAM CONDUCTS RESEARCH OR PROTOTYPES ON A SOLUTION TO PROVE ITS VIABILITY
- DevOps: PRACTICES FOR CREATING A SMOOTH FLOW OF DELIVERING BY IMPROVING COLLABORATION BETWEEN DEVELOPMENT AND OPERATIONS STAFF
- LIFE CYCLE: THE PROCESS THROUGH WHICH A PRODUCT IS

IMAGINED

CREATED

PUT IN USE

MAIN METHODOLOGIES COMMON FEATURES

Rigor without rigidity: how to achieve balance not forgetting that

RIGIDITY IS THE ANTITHESIS OF AGILE

Agile practices build this ability by :

-breaking up large batches of design into much smaller ones

-striving to make the batches as independent as possible

RIGIDITY IS THE ANTITHESIS OF AGILE

BE FLEXIBLE

HEAVYWEIGHT&LIGHTWEIGHT METHODOLOGIES

A ***heavyweight methodology*** is a complex method with many rules

A **lightweight methodology** is a simple method with a few rules.

It emphasizes the need to deal with change in requirements by being **flexible and adaptive**

- User requirements play a key role in both methodologies

HEAVYWEIGHT METHODOLOGIES

Based on a **sequential** series of steps, such as requirements definition, solution build, testing and deployment.

The IT solution isn’t built until the full requirements are determined.

Any team larger than 10-20 people and working in multiple locations is a good candidate for a heavyweight methodology as a tighter control and higher formalization are needed

LIGHTWEIGHT METHODOLOGIES:less documented&more code oriented

Proper change accomodation.

People orientation rather than process orientation

Large use of dynamic checklists.

Small project teams

Fostering knowledge sharing by learning after each iteration issues

Frequent cycles provide more opportunities for clients to review requirements for new needs.

No “heavy” project documentation but focus on the schedule

Focus on value-added releases and addressing architectural risk early

DECISION BETWEEN HEAVYWEIGHT AND LIGHTWEIGHT METHODOLOGIES

A wrong methodology is critical to the project. 10 essential points to be evaluated:

1. Budget
2. Team size
3. Project criticality
4. Technology used
5. Documentation
6. Training
7. Best practices/lessons learned
8. Tools and techniques
9. Existing processes
10. Software type

8 MAIN METHODOLOGIES+*HYBRID MODELS*

So which is best?

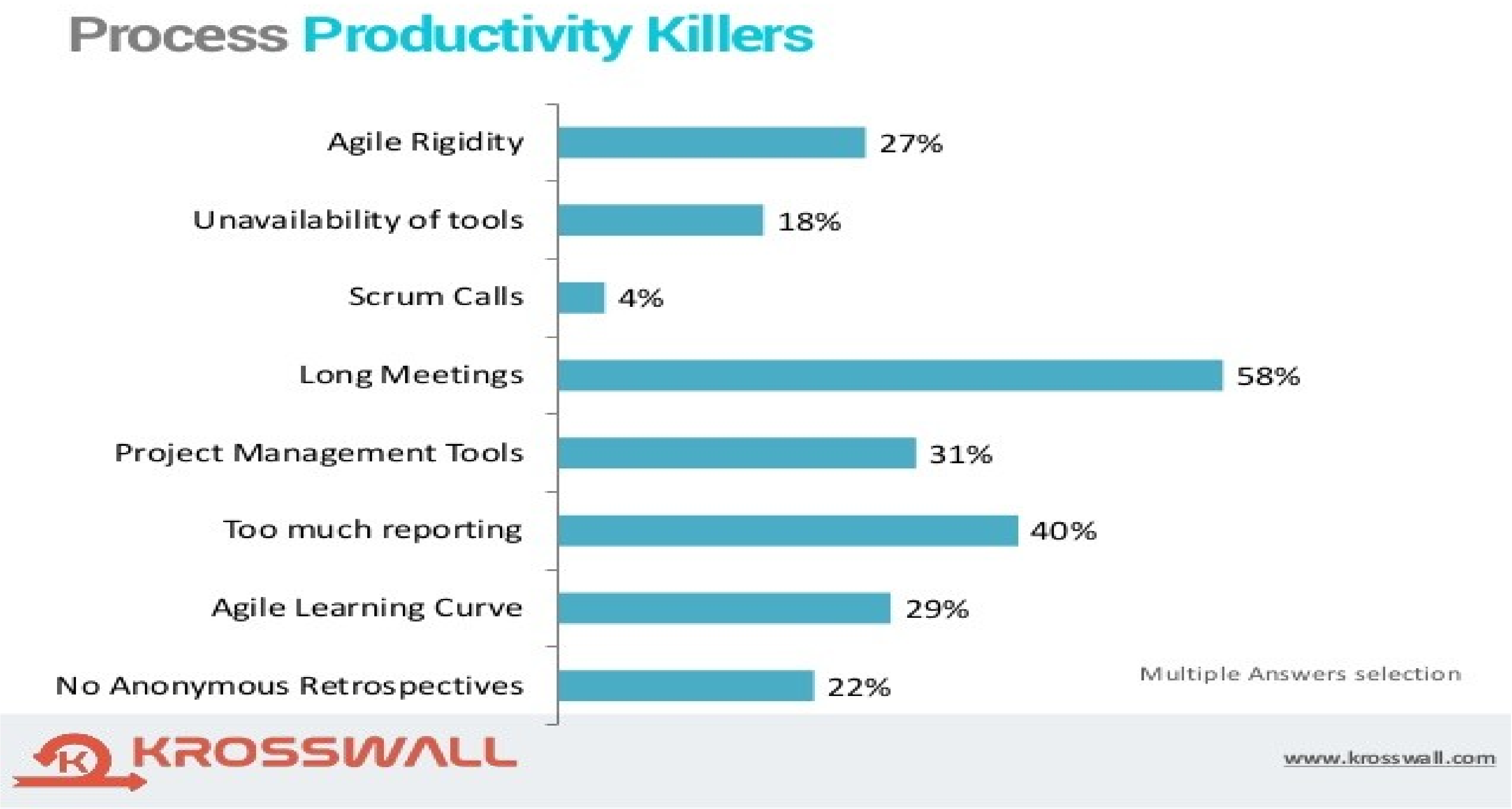
A methodology must be simple, clearly effective, small in terms of required work

The acceptance of a methodology is limited by the group's ability to change its work habits and by its tolerance for seemingly bureaucratic content.

The lightweight methodology approach exemplified by XP,Crystal,Lean,Kanban, Scrum, is more popular for smaller teams

BUT…NEVER FORGET THAT…

GOAL IS PRODUCTIVITY&VALUE



8+1 MAIN METHODOLOGIES

HEAVYWEIGHT (FULL) :

- FDD: FEATURE-DRIVEN DEVELOPMENT
- DSDM:DYNAMIC SYSTEMS DEVELOPMENT METHOD
- AUP

LIGHTWEIGHT :

- XP:EXTREME PROGRAMMING
- LEAN PRODUCT DEVELOPMENT
- KANBAN
- CRYSTAL
- **SCRUM**

HYBRID MODELS

8+1 MAIN METHODOLOGIES_

METHOD	TYPE	KEY POINTS	SPECIAL FEATURES	STRENGHTS	WEAKNESSES
FDD	HEAVYWEIGHT	perspective of features valued by customers	use of cumulative flow diagrams	scalability	poor documentation to the client
DSDM	HEAVYWEIGHT	Attention to early architectural aspects	Business needs	Deliver on time&clear communication	Lack of details for developers
AUP	HEAVYWEIGHT	team know what they're doing.	A development release iteration	Simplicity versus the forefather RUP	Need of expert developers
XP	LIGHTWEIGHT	customer driven	Test-first scheme	Customer involvment	No measure of quality
LEAN	LIGHTWEIGHT	Visual tools	Waste elimination	Quality&speed	success depends on good technical skills
KANBAN	LIGHTWEIGHT	signboard	visualization	flexibility	Lack of timing

FDD FEATURE-DRIVEN DEVELOPMENT

SIMPLE YET POWERFUL APPROACH

FDD IS BASED ON THE **PERSPECTIVE OF FEATURES VALUED BY CLIENTS**

THIS METHODOLOGY POPULARIZED TRACKING AND DIAGNOSTIC TOOLS LIKE **CUMULATIVE FLOW DIAGRAMS** (ONE PAGE SUMMARIES OF PROJECT PROGRESS)

CUMULATIVE FLOW DIAGRAM

It is an area graph that depicts the quantity of work in a given state-

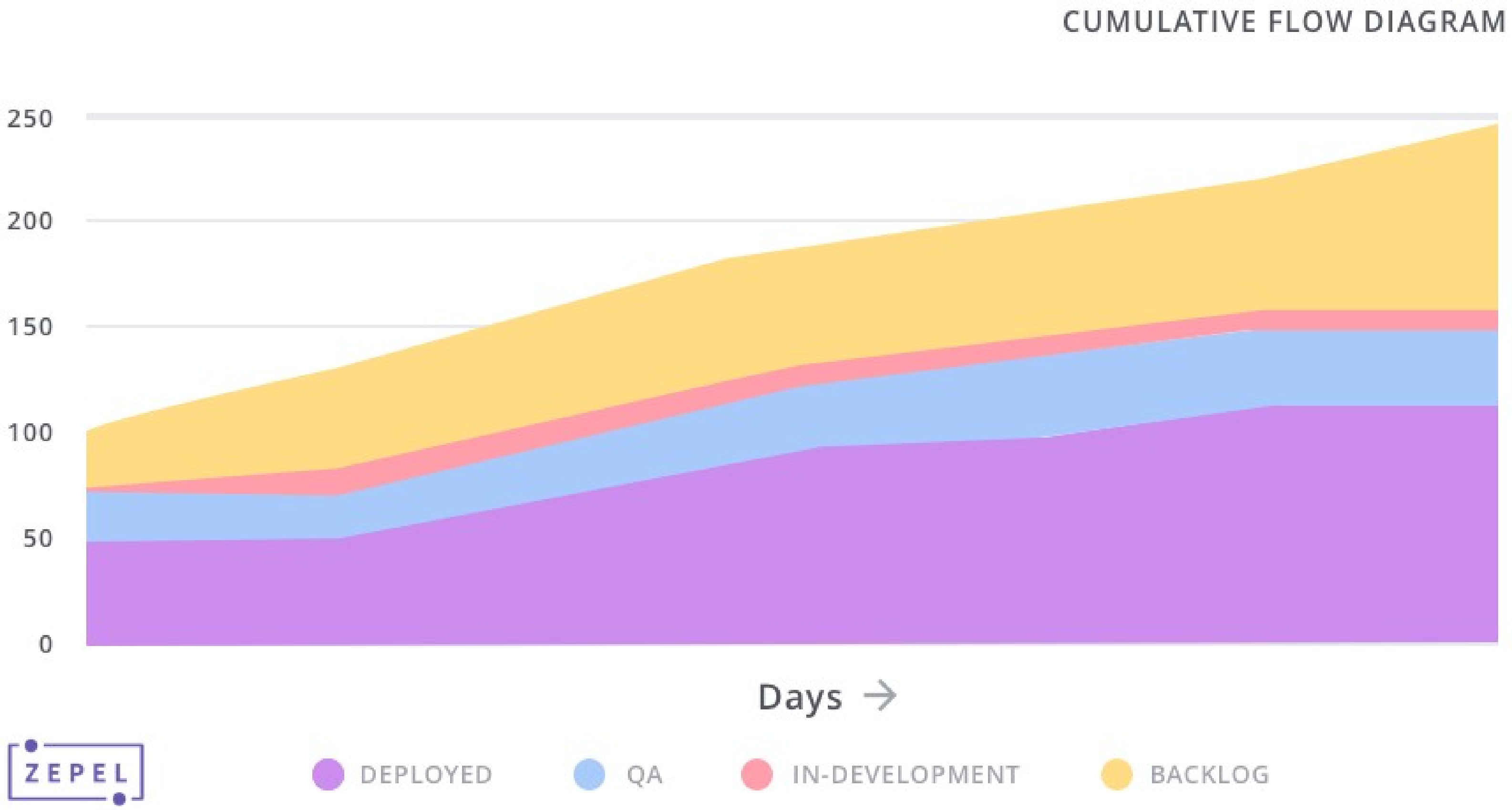
Structured on 2 axis: number of features to be developed and the spent time

4 different relationships are reported:

- **Completed tasks**
- **Under test tasks**
- **Under processing tasks**
- **Not yet started tasks**

CFDs have a strong focus on identifying and rooting out the causes of dramatic changes in throughput.

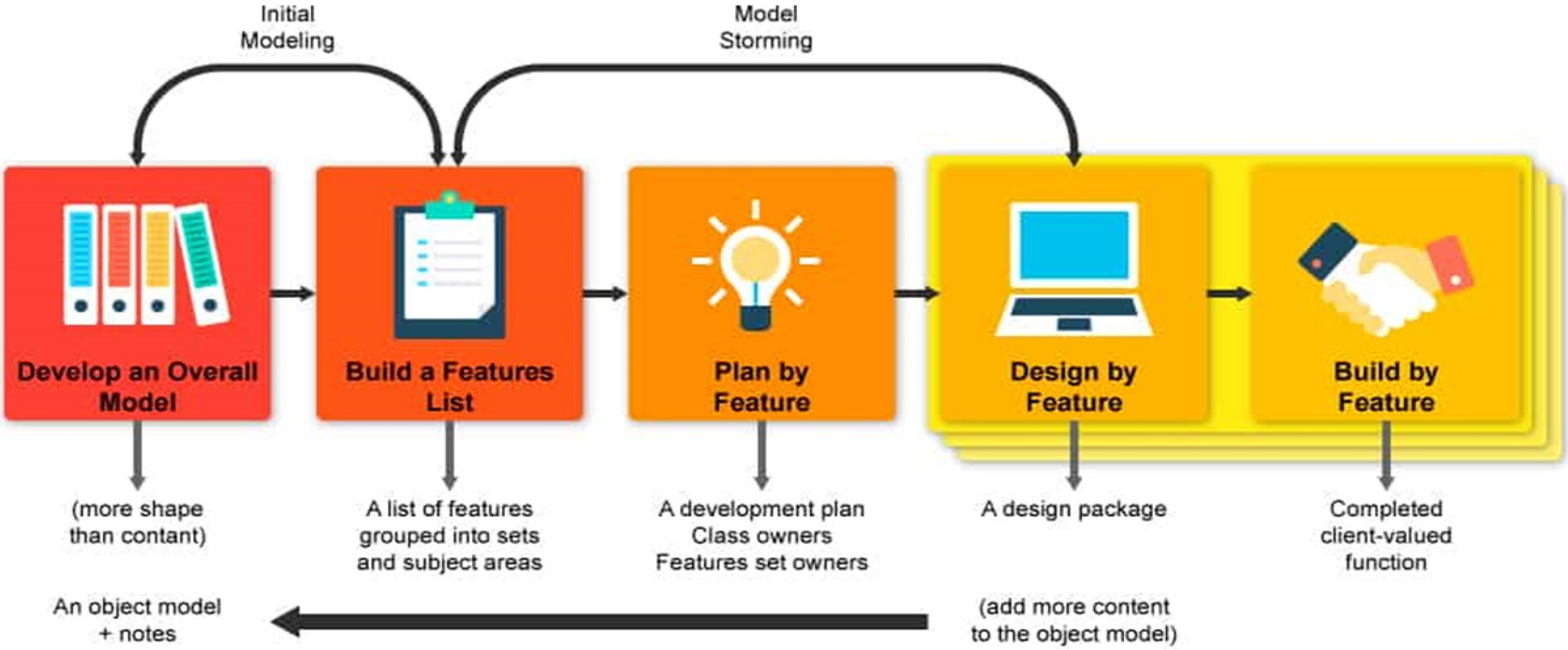
CUMULATIVE FLOW DIAGRAM



FDD FEATURE-DRIVEN DEVELOPMENT

1. DEVELOP A GENERAL MODEL FOR THE PRODUCT
2. BUILD A FEATURES LIST
3. PLAN THE WORK
4. MOVE THROUGH DESIGN BUILDING ITERATIONS BUILD THE FEATURES

FDD FEATURE-DRIVEN DEVELOPMENT



FDD FEATURE-DRIVEN DEVELOPMENT

FDD PRACTICES (FROM SOFTWARE ENGINEERING)

1. DOMAIN OBJECT MODELING
2. DEVELOPING BY FEATURE
3. INDIVIDUAL CLASS (CODE) OWNERSHIP
4. FEATURE TEAMS
5. INSPECTIONS
6. CONFIGURATION MANAGEMENT
7. REGULAR BUILDS
8. VISIBILITY OF PROGRESS AND RESULTS

FDD FEATURE-DRIVEN DEVELOPMENT:1

DOMAIN OBJECT MODELING=EXPLORATION OF BUSINESS ENVIRONMENT OF THE PROBLEM TO BE SOLVED

In software engineering, a **domain model** is a conceptual model of the domain that contains both behaviour and data.

FDD FEATURE-DRIVEN DEVELOPMENT:2

DEVELOPING BY FEATURE=BREAKING FUNCTIONS DOWN INTO TWO WEEKS OR SHORTER CHUNKS OF WORK AND CALLING THEM

FEATURES

FDD FEATURE-DRIVEN DEVELOPMENT:3

INDIVIDUAL CLASS (CODE) OWNERSHIP:

AREAS OF CODE WITH A SINGLE OWNER FOR CONSISTENCY,PERFORMANCE, CONCEPTUAL INTEGRITY

FDD FEATURE-DRIVEN DEVELOPMENT:4

SMALL, DYNAMICALLY FORMED TEAMS THAT REVIEW DESIGNS AND ALLOW MULTIPLE DESIGN OPTIONS TO BE EVALUATED BEFORE A DESIGN IS CHOSEN.

THE GOAL IS RISK MITIGATION OF INDIVIDUAL OWNERSHIP

FDD FEATURE-DRIVEN DEVELOPMENT:5

INSPECTIONS:

REVIEWS THAT HELP ENSURE GOOD-QUALITY DESIGN AND CODE

FDD FEATURE-DRIVEN DEVELOPMENT:6

CONFIGURATION MANAGEMENT:

LABELING CODE,TRACKING CHANGES,MANAGING THE SOURCE CODE

FDD FEATURE-DRIVEN DEVELOPMENT:7

REGULAR BUILDS:

ASSURANCE THAT THE NEW CODE INTEGRATES WITH THE EXISTING ONE

IT ALSO ALLOWS AN EASY CREATION OF DEMO

FDD FEATURE-DRIVEN DEVELOPMENT:8

VISIBILITY OF PROGRESS AND RESULTS=

TRACKING PROGRESS BASED ON THE COMPLETED WORK

FDD FEATURE-DRIVEN DEVELOPMENT

ADVANTAGES	DISADVANTAGES
Very good understanding of project’s scope and context	Not ideal on smaller projects and where there are only a few developers as it is hard to take on various roles.
Fewer meetings. Use of documentation to communicate.	High dependency on a chief programmer who acts as coordinator, lead designer, mentor.
User-centric approach: the client is the end user.	No written documentation to the client, although there is a lot of documented communication among team members.
Works well with large-scale, long-term projects. It is very scalable growing as the project grows. The five steps make it easier to come up to speed	Emphasizes individual code ownership instead of a shared team ownership.
Breaks feature sets into smaller chunks and regular iterative releases, which makes it easier to track and fix coding errors, reduces risk, and allows a quick turnaround	May not work well with older systems because there is already a system in place and no overall model to define it

FDD FEATURE-DRIVEN DEVELOPMENT

WHEN IS IT WORTH TO ADOPT FDD?

This agile methodology is well-suited for long-term projects that continually change and add features in regular, predictable iterations.

DSDM:DYNAMIC SYSTEM DEVELOPMENT METHOD

ONE OF THE EARLIER AGILE METHODS, QUITE PERSPECTIVE AND DETAILED

BROAD COVERAGE OF THE PROJECT LIFE CYCLE,

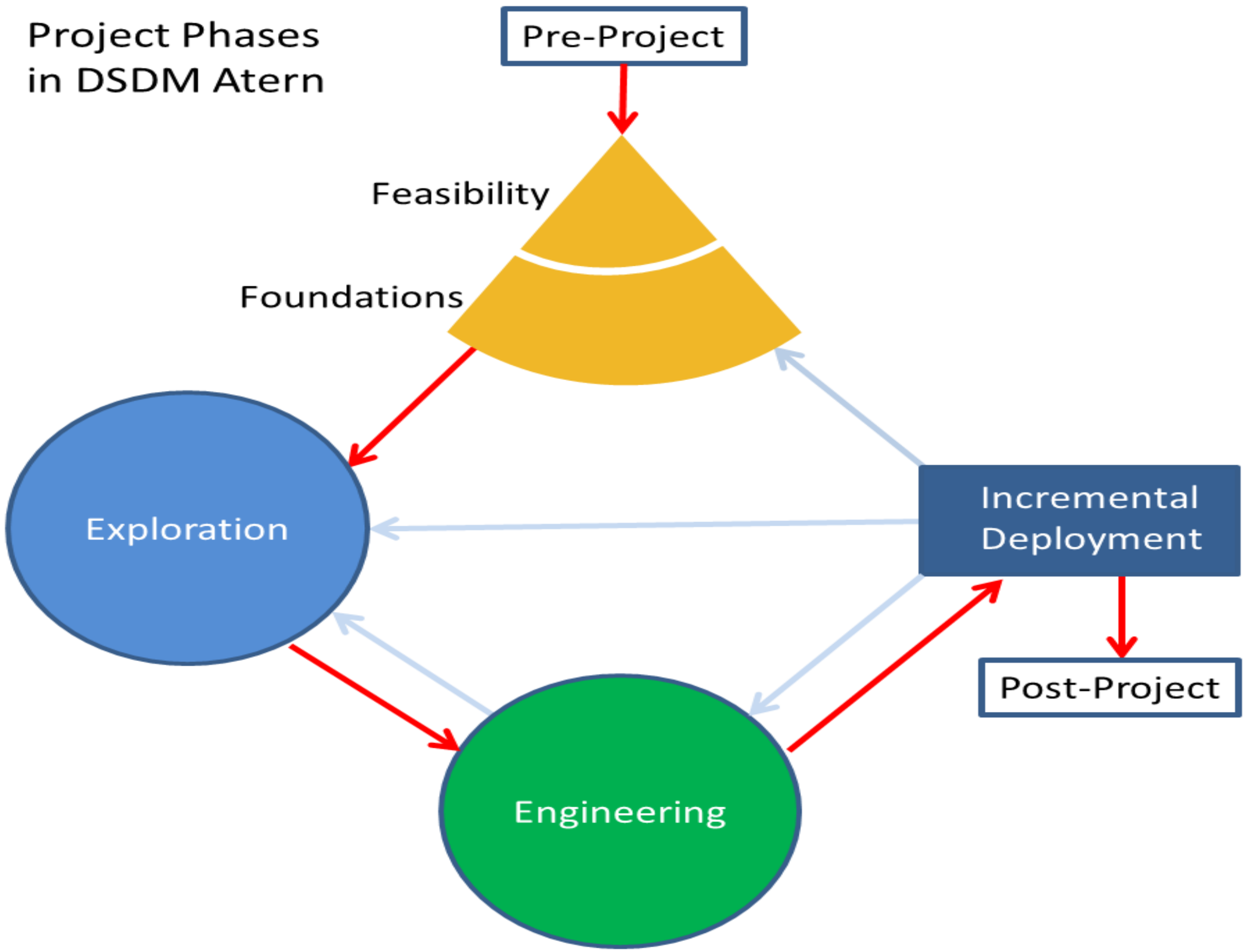
ENCOMPASSING ASPECTS OF AGILITY,

RANGING FROM FEASIBILITY AND BUSINESS CASE TO IMPLEMENTATION THROUGH EXPLORATION AND ENGINNERING

DSDM:DYNAMIC SYSTEM DEVELOPMENT METHOD

METHOD	TYPE	KEY POINTS	SPECIAL FEATURES	STRENGHTS	WEAKNESSES
FDD	HEAVYWEIGHT	perspective of features valued by customers	use of cumulative flow diagrams	scalability	poor documentation to the client
DSDM	HEAVY WEIGHT	Attention to early architect. aspects	Business needs	Deliver on time&clear communication	Lack of details for developers
AUP	HEAVYWEIGHT	team know what they're doing.	A development release iteration	Simplicity versus the forefather RUP	Need of expert developers
XP	LIGHTWEIGHT	customer driven	Test-first scheme	Customer involvment	No measure of quality
LEAN	LIGHTWEIGHT	Visual tools	Waste elimination	Quality&speed	success depends on good technical skills

DSDM LIFE CYCLE



DSDM PRINCIPLES

1. FOCUS ON BUSINESS NEEDS
2. DELIVER ON TIME
3. COLLABORATE
4. NEVER COMPROMISE QUALITY
5. BUILD INCREMENTALLY FROM FIRM FOUNDATIONS
6. DEVELOP ITERATIVELY
7. COMMUNICATE CONTINUOUSLY AND CLEARLY
8. DEMONSTRATE CONTROL

DSDM: THE FOREFATHER

DSDM INFLUENCED THE DEVELOPMENT OF AGILE BY HELPING TO POPULARIZE --**EARLY ARCHITECTURAL CONSIDERATIONS**

-AGILE CONTRACTS

-**AGILE SUITABILITY FILTERS:** tools assessing if an agile approach fits a project, subjective and not predictors of suitability or project success. They should be used as conversation starters and where risks may occur

AUP:AGILE UNIFIED PROCESS

A SIMPLICISTIC AND UNDERSTANDABLE APPROACH TO DEVELOPING BUSINESS APPLICATION SOFTWARE USING AGILE TECHNIQUES AND CONCEPTS

IT IS A SIMPLIFIED VERSION OF THE RATIONAL UNIFIED PROCESS (RUP) THAT CAN NOT BE CLASSIFIED AS AGILE

AUP and RUP

- The AUP is created by the Agile Alliance.
- The RUP is created by Rational Software.

RUP MAIN FEATURES:

- Use-case driven from inception to deployment.
- Architecture-centric as a function of user needs.
- Iterative and incremental:large projects are divided into smaller ones

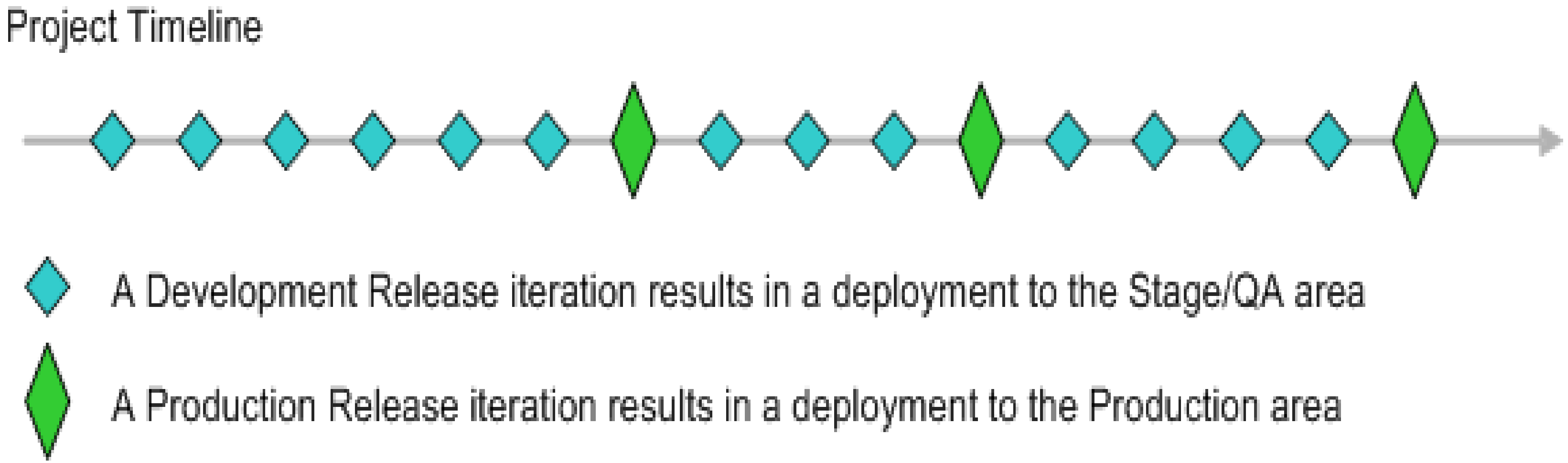
MAIN METHODOLOGIES_

METHOD	TYPE	KEY POINTS	SPECIAL FEATURES	STRENGHTS	WEAKNESSES
FDD	HEAVYWEIGHT	perspective of features valued by customers	use of cumulative flow diagrams	scalability	poor documentation to the client
DSDM	HEAVYWEIGHT	Attention to early architectural aspects	Business needs	Deliver on time&clear communication	Lack of details for developers
AUP	HEAVYWEIGHT	team know what they're doing.	development release iteration	Simplicity versus forefather RUP	Need of expert developers
XP	LIGHTWEIGHT	customer driven	Test-first scheme	Customer involvment	No measure of quality
LEAN	LIGHTWEIGHT	Visual tools	Waste elimination	Quality&speed	success depends on good technical skills
KANBAN	LIGHTWEIGHT	signboard	visualization	flexibility	Lack of timing
CRYSTAL	LIGHTWEIGHT	Family of custom methods	People focus	Frequent deliveries	Only small teams

AUP:AGILE UNIFIED PROCESS

- The Agile Unified Process distinguishes between two types of iterations.
- A development release iteration results in a deployment to the quality-assurance and/or demo area.
- A production release iteration results in a deployment to the production area. This is a significant refinement to the Rational Unified Process.

The Agile Unified Process distinguishes between two types of iterations.



A Development Release iteration results in a deployment to the Stage/QA area

A Production Release iteration results in a deployment to the Production area

AUP:AGILE UNIFIED PROCESS

MAIN PRACTICES:

1. test-driven development (TDD)
2. agile modeling (AM)
3. agile change management
4. database refactoring to improve productivity.

AUP:AGILE UNIFIED PROCESS

1. test-driven development (TDD)

A software development process based on the repetition of a very short development cycle with requirements turned into specific test cases.

After code efficient improvement tests pass.

Opposed to software development allowing code addition even if not proven to meet requirements.

AUP:AGILE UNIFIED PROCESS

2. agile modeling (AM) :methodology to document and model software

3. Document continuously and as late as possible
4. Requirements are specified in the form of executable "customer tests", instead of non-executable "static" documentation.
5. Information (models, documentation, software), is stored in one place only
6. Light-weight, high-level modeling,just barely good enough (JBGE)
7. Prioritized requirements

AUP:AGILE UNIFIED PROCESS

3. database refactoring to improve productivity

Database refactoring does not change the way data is interpreted or used and does not fix bugs or add new functionality.

Refactoring a database is performed when it is requested:

-To develop the schema in an evolutionary manner in parallel with the evolutionary design of the rest of your system.

-To fix design problems with an existing legacy database schema.

-To implement what would be a large change as a series of small, low-risk changes.

AUP:AGILE UNIFIED PROCESS

4. agile change management

Change management is a structured approach for people and organizations which fosters transiction from a current arrangement to a new desired one.

It’s a new concept that carries great benefits to organizations

A combination of agile with change management get the best of both worlds.

-Determine the Scope of the Change and of Incorporating the Change

-Gain Approval or Rejection of the Change. ...

-Communicate and Implement an Approved Change Request. ...

Manage Change or It Will Manage You!

3nd LESSON (part 1):8 MAIN METHODOLOGIES

XP:EXTREME PROGRAMMING

FOCUS:

SOFTWARE DEVELOPMENT BEST PRACTICES

3nd LESSON (part 1):8 MAIN METHODOLOGIES

XP:EXTREME PROGRAMMING (1)

- CORE VALUES:

-SIMPLICITY:FIND THE SIMPLEST ABLE TO WORK THING AND BUILD THE SOLUTION FIRST

-COMMUNICATION: MAKE SURE ALL THE TEAM MEMBERS KNOW WHAT IS EXPECTED OF THEM AND WHAT OTHERS ARE WORKING ON

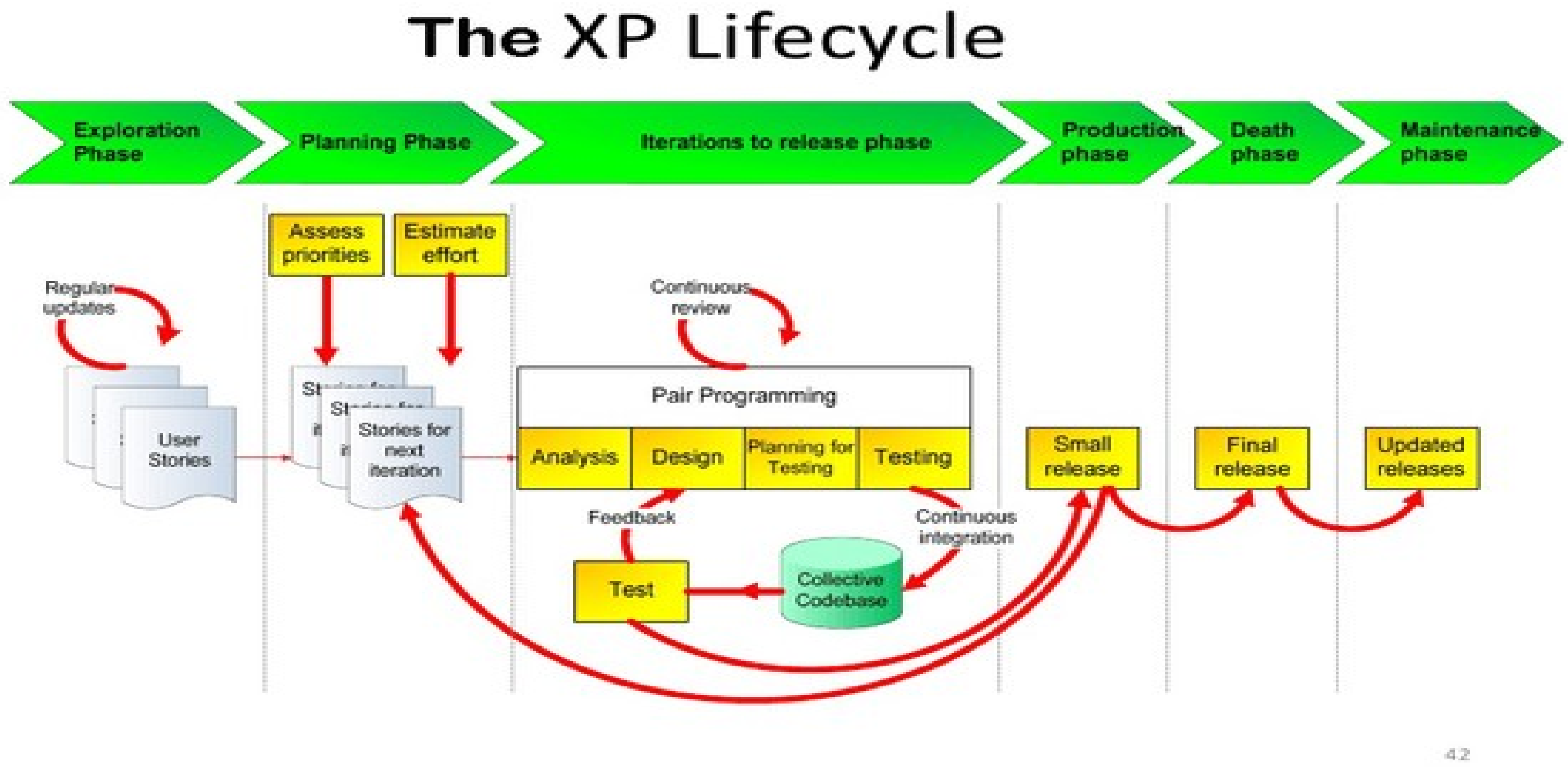
-FEEDBACK: GET IMPRESSIONS OF SUITABILITY EARLY BY THE TEAM

-COURAGE: ALLOW WORK ENTIRELY VISIBLE TO OTHERS AND SHARE CODE

-RESPECT: THROUGH PAIR PROGRAMMING MEMBERS RECOGNIZE WORKING DIFFERENTLY AND RESPECT DIFFERENCES

3nd LESSON (part 1):8 MAIN METHODOLOGIES

XP:EXTREME PROGRAMMING (2)



XP:EXTREME PROGRAMMING (3)

- XP teams use lightweight requirements («user stories») to plan releases and iterations
- Iterations are generally 2 weeks long and developers work in pair writing code with frequent and rigorous testing
- Then, after approval of customers,software is delivered as small releases

3nd LESSON (part 1):8 MAIN METHODOLOGIES

XP:EXTREME PROGRAMMING (4)

XP TEAM ROLES

COACH= mentor, facilitator helping the team become more effective

CUSTOMER= business representative providing requirements and priorities and confirms that the product works as intended

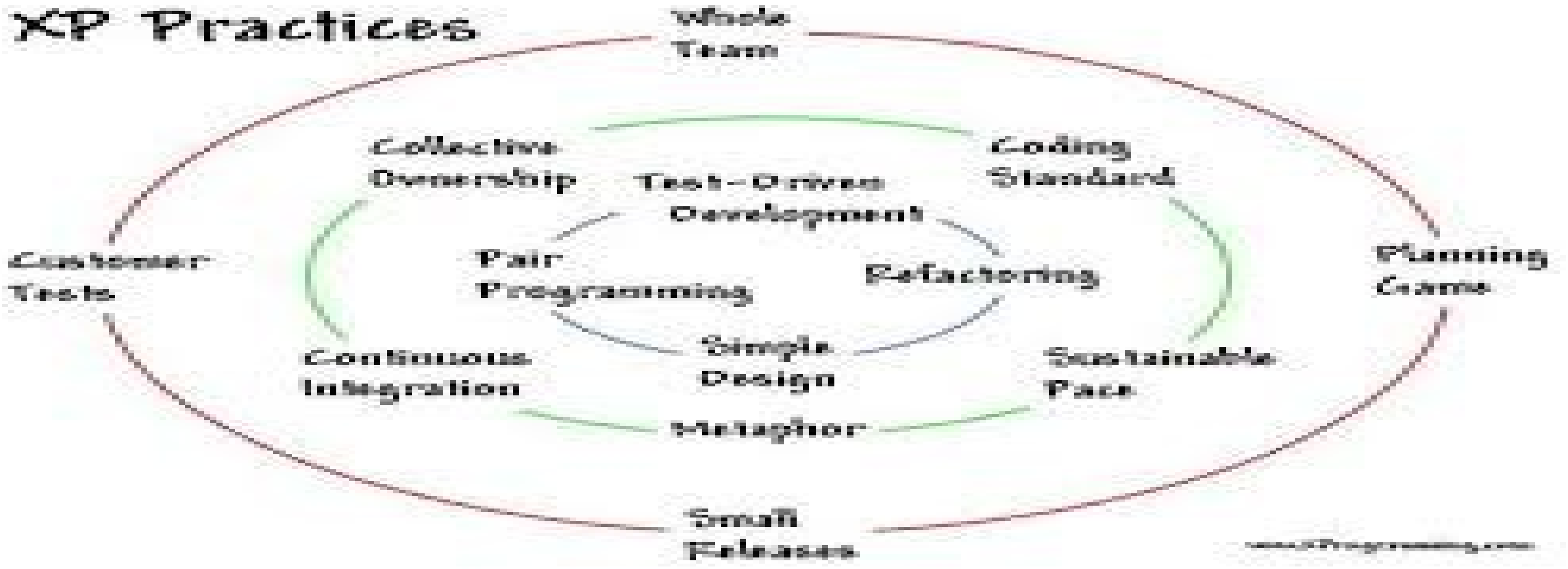
PROGRAMMERS=developers of the code for the requested user stories

TESTERS=providers of quality assurance also supporting the customer to define acceptance tests for the user stories

3nd LESSON (part 1):8 MAIN METHODOLOGIES

XP:EXTREME PROGRAMMING (6)

XP PRACTICES



3nd LESSON (part 1):8 MAIN METHODOLOGIES

XP:EXTREME PROGRAMMING (7)

1. WHOLE TEAM
2. PLANNING GAMES
3. SMALL RELEASES
4. CUSTOMER TESTS
5. COLLECTIVE CODE OWNERSHIP
6. CODE STANDARDS
7. SUSTAINABLE PACE
8. METAPHOR
9. CONTINUOUS INTEGRATION
10. TEST DRIVEN DEVELOPMENT
11. REFACTORING
12. SIMPLE DESIGN
13. PAIR PROGRAMMING

3nd LESSON (part 1):8 MAIN METHODOLOGIES

XP:EXTREME PROGRAMMING (8)

When Is Xp Not Appropriate

- Concurrent middleware development. Here the enormous number of existing usage scenarios combined with the impossibility of reliably unit testing for correct concurrency behaviour renders XP inapplicable. CRC cards, 'low-frequency' refactoring, and possibly some other practices adopted by XP do apply, but not the whole of XP.
- OS kernels and device drivers. Similar rationale to concurrent middleware.
- Safety critical systems where change has to be managed very carefully to preserve safety. Remember "Testing can only show the presence of errors, not their absence."
- 'Legacy' systems where the volume of code far outstrips the time available to maintain it but where 'tinkering' is sometimes necessary. Unpleasant yes, but sometimes necessary.
- When the whole project is making expensive-to-change decisions based on the software (e.g. changing (refactoring) an application that works well on distributed boxes to one that requires one big box after the hardware has been bought). Most of these probably change the SystemMetaphor too, though.

3nd LESSON (part 1):8 MAIN METHODOLOGIES

KANBAN (=signboard) (1)

- DERIVED FROM THE LEAN PRODUCTION SYSTEM
- A BOARD SHOWS THE WORK ITEMS IN EACH STAGE OF A PRODUCTION PROCESS

TO DO	IN PROGRESS	DONE
AMEND ORDER	CREATE ORDER	DATABASE SCHEMA
ITEM DETAILS	STOCK SEARCH	LOGIN
ARCHIVE ORDER	STOCK UPDATE	
ORDER REFUND	PROCESS ORDER	

3nd LESSON (part 1):8 MAIN METHODOLOGIES

KANBAN (=signboard) (2)

5 PRINCIPLES

1. VISUALIZE THE WORKFLOW
2. LIMIT WIP (WORK INPROGRESS)
3. MANAGE FLOW
4. MAKE PROCESS POLICIES EXPLICIT
5. IMPROVE COLLABORATIVELY

3nd LESSON (part 1):8 MAIN METHODOLOGIES

KANBAN (=signboard) (3)

DISTINCT DIFFERENTIATING FEATURE:

«PULL SYSTEM EMPLOYMENT» (LESS EMPHASIS ON ITERATIONS)

IT MEANS WORK MOVEMENT THROUGH THE DEVELOPMENT PROCESS RATHER THAN PLANNING IN TIMEBOXED ITERATIONS

EACH TIME A KANBAN TEAM COMPLETES A WORK ITEM, IT TRIGGERS A «PULL» TO GO ON INTO THE NEXT ITEM

BUT ONLY SOMEF SLOTS ARE AVAILABLE AND WHENEVER THERE IS AN EMPTY SLOT IT IS A SIGNAL TO PULL WORK FROM PREVIOUS STAGES

3nd LESSON (part 1):8 MAIN METHODOLOGIES

KANBAN (=signboard) (4)

WIP LIMITS

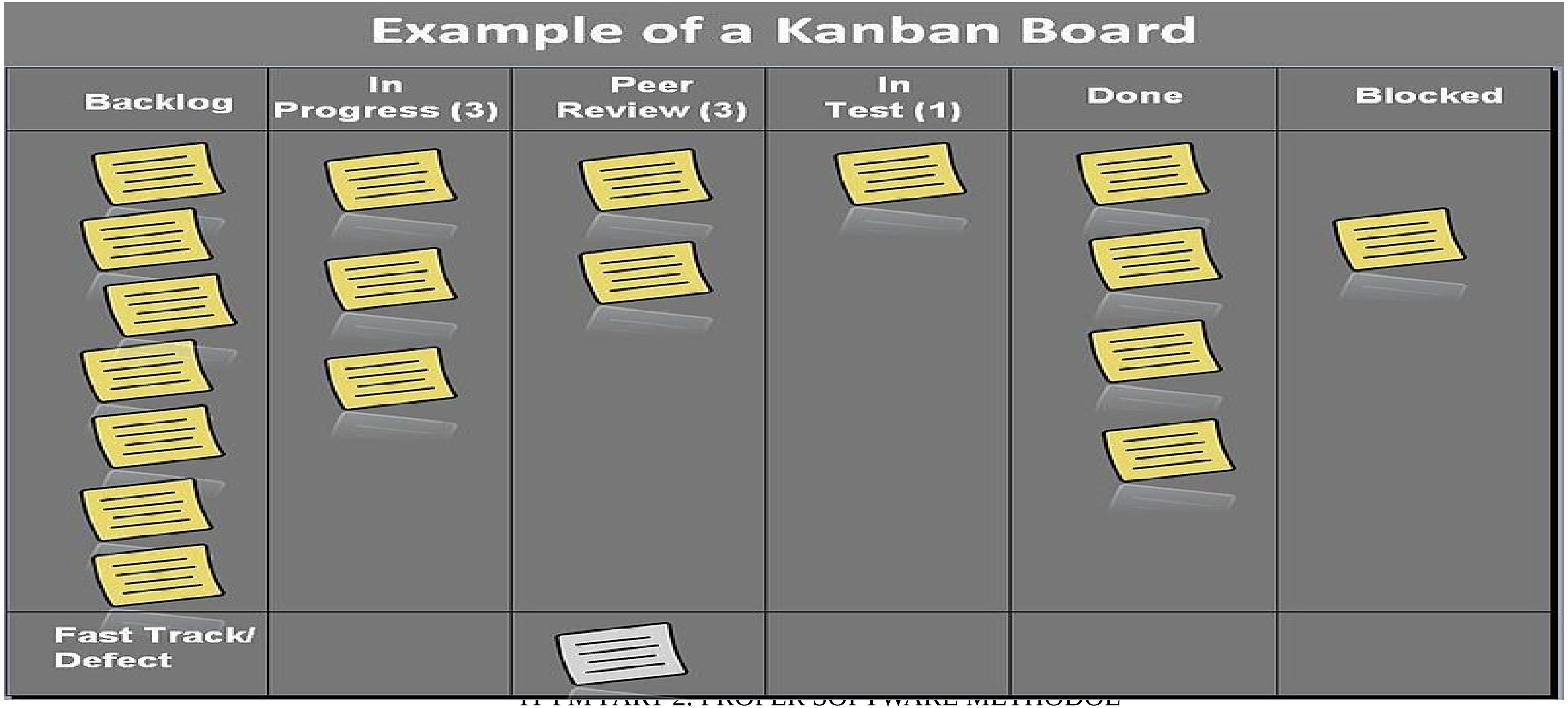
CAPPING THE NUMBER OF ITEMS THAT CAN BE IN A GIVEN STATE OF PROGRESS

THIS IS DEFINED BY THE COLUMNS IN THE KANBAN BOARD: ONCE THE LIMIT AT THE TOP OF A COLUMN IS REACHED, NO NEW ITEMS MAY BE MOVED INTO THAT COLUMN UNTIL ANOTHER ITEM IS MOVED OUT

3nd LESSON (part 1):8 MAIN METHODOLOGIES

KANBAN (=signboard) (5)

- KANBAN BOARD WITH WIP LIMITS



3nd LESSON (part 1):8 MAIN METHODOLOGIES

KANBAN (=signboard) (6)

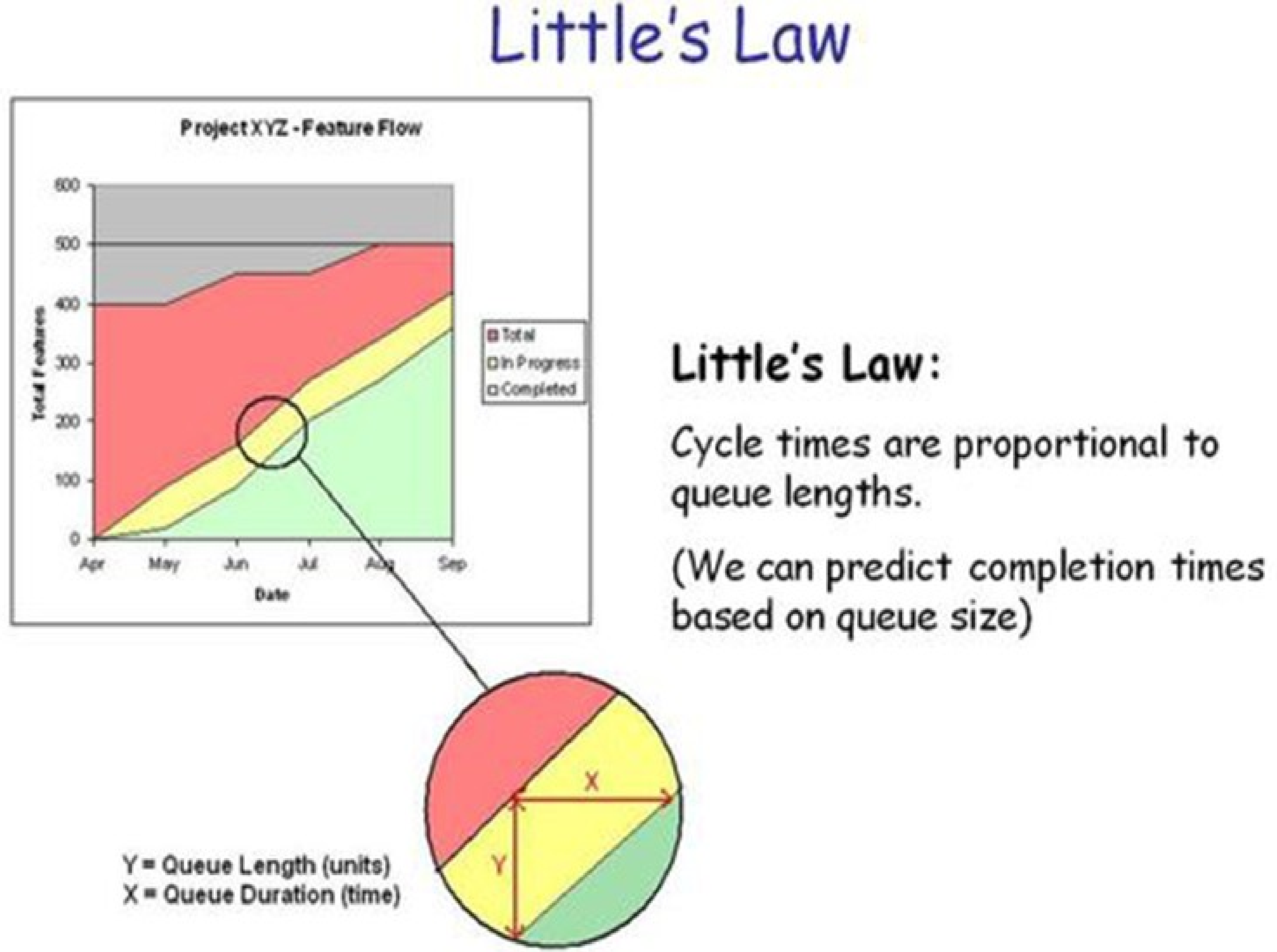
IMPORTANCE OF WIP LIMITS

LOWERNG WIP INCREASES TEAM’S PRODUCTIVITY AND SPEEDS UP THE RATE AT WHICH THE WORK IS COMPLETED AS SHOWN BY

LITTLE’S LAW

3nd LESSON (part 1):8 MAIN METHODOLOGIES

KANBAN (=signboard) (7)



Little's Law

Little's Law:

Cycle times are proportional to queue lengths.

(We can predict completion times based on queue size)

Y = Queue Length (units)
X = Queue Duration (time)

UGIES

3nd LESSON (part 2):8 MAIN METHODOLOGIES

CRYSTAL (1)

NOT A METHOD BUT A FAMILY OF SITUATIONAL&CUSTOMIZED ONES

METHODS ARE CODED BY COLOR NAMES AND ARE CUSTOMIZED BY CRITICALITY AND TEAM SIZE WHICH ALLOWS CRYSTAL TO COVER A WIDE RANGE OF PROJECTS FROM A SMALL TEAM BUILDING A LOW CRITICALITY SYSTEM TO A LARGE TEAM BUILDING A HIGH-CRITICALITY SYSTEM

REVIEW: WHAT DID I TRY TO COMMUNICATE?

- MANY METHODOLOGIES WITHIN SOFTWARE PROJECTS
- FOCUS ON AGILE, AS MORE DIFFERENT VERSUS TRADITIONAL WATERFALL
- HEAVYWEIGHT (MORE RIGID) and LIGHT WEIGHT
- HEAVYWEIGHT:

-FDD

-DSDM

-AUP

PROJECT MANAGEMENT FOR IT
REVIEW LESSONS 1-3

AGILE✉ITERATION&REQUIREMENT

↓

«DONE»

METHODOLOGIES to get VALUE

PROJECT MANAGEMENT FOR IT REVIEW LESSON 3

- **FDD✉model, features, cumulative flow diagrams**
- **DSDM✉feasibility,functionality,business needs**
- **AUP✉Simplicity (just development and producion release) versus the forefather RUP (***Architecture-centric*)
- **KANBAN**
- **CRYSTAL**
- **HYBRID**

PROJECT MANAGEMENT FOR IT REVIEW OF LESSON 1,2,3

- INNOVATION
- METHODOLOGY
- REQUIREMENTS
- ITERATION
- VALUE DRIVEN

THANKS

NEXT: LESSON 4

SCRUM