

PMI-ACP On-Demand Course Slides

*Course ID: PMTRAINING-ACP35-V6
Contact Hours / PDUs: 21*

For more information on project management certification training, please visit www.PMTraining.com

ACP Exam Prep

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

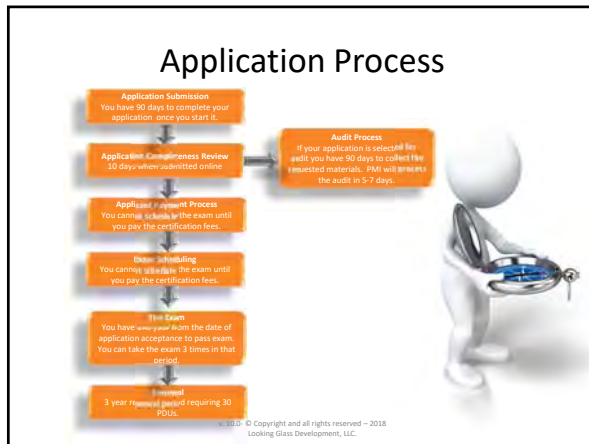
Agenda

- The Process
- The Exam
- Agile Principles & Mindset
 - Basics
 - Scrum
 - XP
 - FDD
 - DSDM
 - Crystal
 - Lean
 - Kanban
- Value Driven Delivery
- Stakeholder Engagement
- Boosting Team Performance
- Adaptive Planning
- Problem Detection & Resolution
- Continuous Improvement

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

The Process

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.



ACP Qualifications

Education	+	General Project Experience	+	Agile Project Experience	+	Training in Agile Practices
High School Diploma, Associates Degree or equivalent		2,000 hours (12 months) working on projects in the last 5 years. Hours must be non-overlapping.		1,500 hours (8 months) working on project teams using agile methodologies in the last 3 years in addition and separate from the 2,000 hours of general PM experience.		21 contact hours in agile practices.
PMs and PgMPs will not have to prove their general project management experience.						

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Scheduling Your Exam

- Online: <http://www.prometric.com/pmi>
 - Select schedule an appointment.
 - Choose your country & state.
 - Create your account & log in.
 - Select PMI-Project Management Institute in the client field & PMI-Project Management Institute (PR0, PR1) in the program field.
 - Select CA0-002 PMI-Agile Certified Practitioner (PMI-ACP®) in the exam field.
 - Select an exam site & schedule appointment.
 - Enter your PMI eligibility ID & confirm your e-mail address.
- Call +1 800-268-2802

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Rescheduling/Cancelling

- Within 30 days \$70 charge.
- Within 2 days forfeit entire fee.
- “Extenuating Circumstances”
 - Medical emergency
 - Military deployment
 - Death in immediate family
 - Illness in immediate family
 - Natural disaster
- Work related circumstances will not be accepted.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Fees

- 1st Take Fees
 - PMI Members: \$435
 - Non-PMI Members: \$495
- Retakes
 - PMI Members: \$335
 - Non-PMI Members \$395
- Refunds: \$200 if test not schedule or taken within exam period. No refunds if never take test.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

The Exam

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Exam Results

- An overall pass/fail score.
- Each topic area is assigned one of three levels of proficiency.
 - Proficient
 - Moderately Proficient
 - Below Proficient

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

The Exam

- Total Questions: 120
- Scored Questions: 100
- 3 Hours to take the exam.
- No scheduled breaks, but you are allowed one bio break.
- Exam preceded by an optional tutorial.
- Exam followed by optional survey.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Exam Breakdown

	Percentage of Questions
Agile Tools & Techniques	50%
Agile Knowledge & Skills	50%
Total:	100%

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

The Exam

Tools and Techniques 50% of Exam

Communications	Including but not limited to: information radiator, team space, agile tooling, osmotic communications for colocated and/or distributed teams, daily stand-ups
Planning, Monitoring & Adapting	Including but not limited to: retrospectives, Scrum/Kanban boards, timeboxing, iteration and release planning, WIP limits, burn down/up charts, cumulative flow diagrams, process tailoring.
Agile Estimation	Including but not limited to: relative sizing/story points, wide band Delphi/planning poker, affinity estimating, ideal time.
Agile Analysis and Design	Including but not limited to: product roadmap, user stories/backlog, story maps, progressive elaboration, wireframes, chartering, personas, agile modeling
Product Quality	Including but not limited to: frequent verification and validation, test-driven development/test first development, acceptance test-driven development, definition of done, continuous integration.
Soft Skills Negotiation	Including but not limited to: emotional intelligence, collaboration, adaptive leadership, negotiation, conflict resolution, servant leadership.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

The Exam

Tools and Techniques 50% of Exam

Value-based Prioritization	Including but not limited to: return on investment (ROI)/net present value (NPV)/internal rate of return (IRR), compliance, customer-valued prioritization, minimally marketable feature (MMF), relative prioritization/ranking
Risk Management	Including but not limited to: risk-adjusted backlog, risk burn down graphs, risk-based spike
Metrics	Including but not limited to: velocity, cycle time, earned value management (EVM) for agile projects, escaped defects
Value Stream Analysis	Including but not limited to: value stream mapping

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Test Breakdown

Knowledge & Skills 50% of Exam

Level	% of Knowledge & Skill Content / % of Exam
Level 1 (18 knowledge/skills)	65% / 33%
Level 2 (12 knowledge/skills)	25% / 12%
Level 3 (13 knowledge/skills)	10% / 5%

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Level 1 (33% of Total Examination Questions)

- Active listening
- Agile Manifesto values & principles
- Assessing & incorporating community & stakeholder values
- Brainstorming techniques
- Building empowered teams
- Coaching & mentoring within teams
- Communications management
- Feedback techniques for product (e.g. prototyping, simulation, demonstrations, evaluations)
- Incremental delivery
- Knowledge sharing
- Leadership tools & techniques
- Prioritization
- Problem-solving strategies, tools, & techniques
- Project & quality standards for Agile projects
- Stakeholder management
- Team motivation
- Time, budget, & cost estimation
- Value-based decomposition & prioritization

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Level 2 (12% of Total Examination Questions)

- Agile frameworks & terminology
- Building high-performance teams
- Business case development
- Colocation (geographic proximity) / distributed teams
- Continuous improvement processes
- Elements of a project charter for an Agile project
- Facilitation methods
- Participatory decision models (e.g. input-based, shared collaboration, command)
- PMI's code of Ethics & Professional Conduct
- Process analysis techniques
- Self assessment
- Value-based analysis

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Level 3 (5% of Total Examination Questions)

- Agile contracting methods
- Agile project accounting principles
- Applying new Agile practices
- Compliance (organization)
- Control limits for Agile projects
- Globalization, culture, & team diversity
- Agile games
- Principles of systems thinking (e.g. complex adaptive, chaos)
- Regulatory compliance
- Variance & trend analysis
- Variations in Agile methods & approaches
- Vendor management

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Key Readings

- Agile Retrospectives: Making Good Teams Great, Esther Derby, Diana Larsen, Ken Schwaber.
- Agile Software Development: The Cooperative Game – 2nd Edition, Alistair Cockburn.
- The Software Project Manager’s Bridge to Agility, Michele Sliger & Stacia Broderick.
- Coaching Agile Teams, Lyssa Adkins.
- Agile Project Management: Creating Innovative Products – 2nd Edition, Jim Highsmith.
- Becoming Agile: ...in an Imperfect World, Greg Smith & Ahmed Sidky.
- Agile Estimating and Planning, Mike Cohn.
- The Art of Agile Development, James Shore.
- User Stories Applied: For Agile Software Development, Mike Cohn.
- Agile Project Management with Scrum, Ken Schwaber.
- Lean-Agile Software Development: Achieving Enterprise Agility, Alan Shalloway, Guy Beaver, James R. Trott.
- Effective Project Management: Traditional, Agile, Extreme, Robert Wysocki.
- Exploring Scrum: The Fundamentals, Dan Rawsthorne with Doug Shimp.
- Kanban In Action, Marcus Hammarberg, Joakim Sunden.
- Kanban: Successful Evolutionary Change for Your Technology Business, David J. Anderson.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Domain Breakdown

Domain	% of Items on Exam
1. Agile Principles & Mindset	16%
2. Value-Driven Delivery	20%
3. Stakeholder Engagement	17%
4. Team Performance	16%
5. Adaptive Planning	12%
6. Problem Detection & Resolution	10%
7. Continuous Improvement (Product, Process, People)	9%

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Agile Principles & Mindset

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Agile Principles & Mindset

Domain Tasks

1. Advocate for agile principles by modeling those principles & discussing agile values in order to develop a shared mindset across the team as well as between the customer & the team.
2. Help ensure that everyone has a common understanding of the values & principles of agile & a common knowledge around the agile practices & terminology being used in order to work effectively.
3. Support change at the system or organization level by educating the organization & influencing processes, behaviors, & people in order to make the organization more effective & efficient.
4. Practice visualization by maintaining highly visible information radiators showing real progress & real team performance in order to enhance transparency & trust.
5. Contribute to a safe & trustful team environment by allowing everyone to experiment & make mistakes so that each can learn & continuously improve the way he or she works.
6. Enhance creativity by experimenting with new techniques & process ideas in order to discover more efficient & effective ways of working.
7. Encourage team members to share knowledge by collaborating & working together in order to lower risks around knowledge silos & reduce bottlenecks.
8. Encourage emergent leadership within the team by establishing a safe & respectful environment in which new approaches can be tried in order to make improvements & foster self-organization & empowerment.
9. Practice servant leadership by supporting & encouraging others in their endeavors so that they can perform at their highest level & continue to improve.

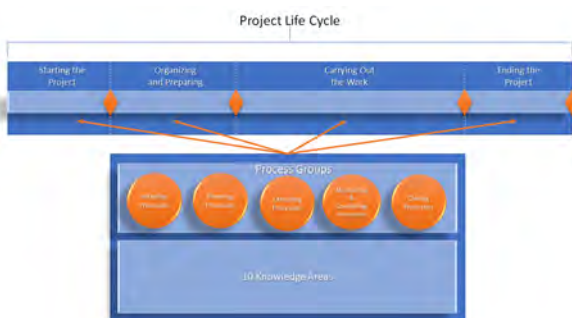
v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

PMBOK Guide vs. Agile

- PMBOK® Guide is the umbrella framework.
- Agile represents a family of lifecycles sometimes called methodologies or frameworks.
- Agile methodologies fit within the PMBOK® Guide.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

The Basics of PM



v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Two Types of Agile

- Iteration-based Agile
 - Team works in iterations to deliver features.
 - Works from most important feature to least.
 - Team does NOT address all the features in an iteration at once.
- Flow-based Agile
 - Team pulls features from the backlog based on its capacity to start work rather than on an iteration-based schedule.
 - Team defines its workflow with columns on a task board and manages the work in progress for each column.
- Agile life cycles are those that fulfill the principles of the Agile Manifesto.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

The Four Types of Life Cycles

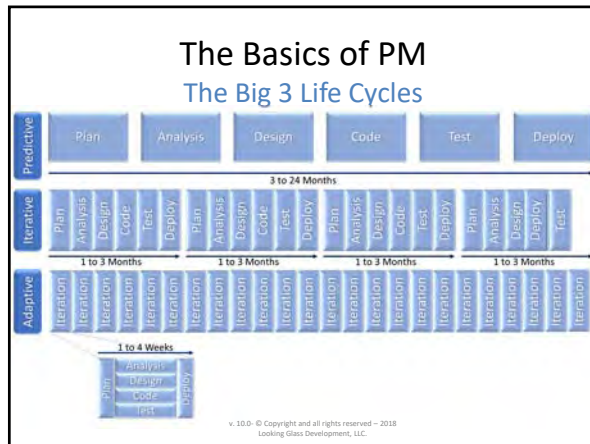
- **Predictive life cycle** – The “traditional” approach. It is a linear process with the majority of planning upfront then executed in a single sequential pass. They take advantage of things that are known and proven. The plan drives the work. Value is only delivered at the end.
- **Iterative life cycle** – An approach that allows feedback for unfinished work to improve and modify future outcomes. Prototypes and proofs are planned, and the outputs are intended to modify the plans at the beginning. When complexity is high or when there are frequent changes or scope is unknown.
- **Incremental life cycle** – An approach that provides finished deliverables in steps that the customer may use immediately. Here we plan to deliver successive subsets of the overall project. The team may deviate from the original vision as it uncovers hidden or misunderstood requirements.
- **Agile life cycle** – An approach that is **BOTH** iterative and incremental to refine work items **AND** deliver frequently. Here we plan and re-plan as more information becomes available.

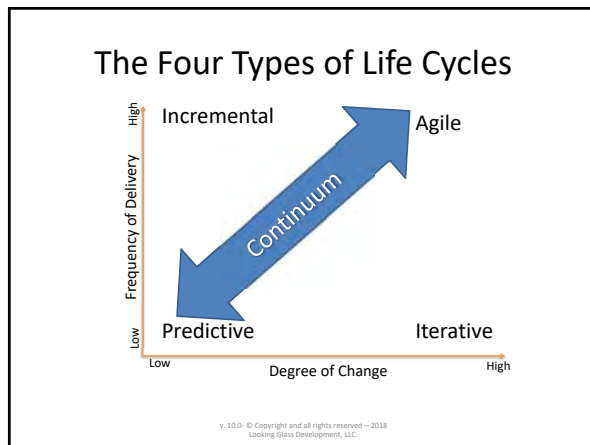
v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

The Four Types of Life Cycles

Characteristics				
Approach	Requirements	Activities	Delivery	Goal
Predictive	Fixed	Performed once for the entire project	Single delivery	Manage costs
Iterative	Dynamic	Repeated until correct	Single delivery	Correctness of solution
Incremental	Dynamic	Performed once for a given increment	Frequent smaller deliveries	speed
Agile	Dynamic	Repeated until correct	Frequent smaller deliveries	Customer value via frequent delivery and feedback

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

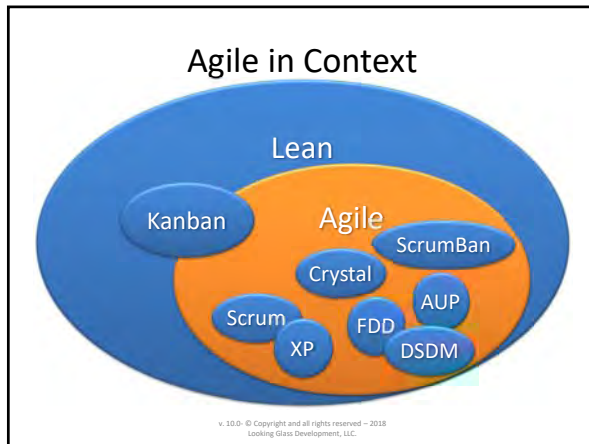




Strategies to Implement Agile

- Adopt a formal agile approach. Learn the approach in detail & then tailor it.
- Implement changes to project practices that fits the project context to get progress on a core value or principle.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.



Agile Methodologies

- 16 different frameworks/methodologies are “Agile”
- Most common include:
 - Scrum
 - Extreme Programming
 - Feature-Driven Development (FDD)
 - Dynamic Systems Development Method (DSDM)
 - Crystal (Crystal Clear, Crystal Yellow, Crystal Orange...)
- Closely related concepts include:
 - Lean software development
 - Kanban
- Exam focuses on Scrum & XP although others might appear.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Overview

Agile is Iterative & Incremental

- **Incremental development** – is a staging & scheduling strategy in which the various parts of the system are developed at different times or rated and integrated as they are completed.
- **Iterative development** – is a rework scheduling strategy in which time is set aside to revise and improve parts of the system.

Alistair Cockburn

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Iterative & Incremental Approaches

- Very short feedback loops
- Frequent adaptation of process
- Reprioritization
- Regularly updated plans
- Frequent delivery

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

The Effects of WIP & “Best Resourcing”

Deliverable X

Option 1	A	A
Option 2	A	C

Deliverable Y

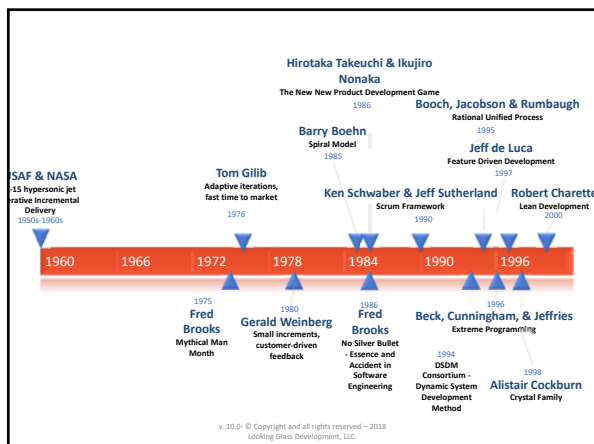
Option 1	B	B	
Option 2	B	B	C

Deliverable Z

Option 1	C	C
Option 2		

- In Option 1 the best resource for the deliverable attempts to do each task for the feature and nothing gets delivered.
- In Option 2 resourcing is applied based upon availability and two features are delivered.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.



Overview The Beginning of Agile

- February, 2001 in Snowbird, Utah

Kent Beck	Mike Beedle
Arie van Bennekum	Alistair Cockburn
Ward Cunningham	Martin Fowler
James Grenning	Jim Highsmith
Andrew Hunt	Ron Jeffries
Jon Kern	Brian Marick
Robert C. Martin	Steve Mellor
Ken Schwaber	Jeff Sutherland
Dave Thomas	

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Overview

Agile Development Values...

- Individuals & Interactions **OVER** Processes & Tools
- Customer Collaboration **OVER** Contract Negotiations
- Responding to Change **OVER** Following a Plan
- Working Software **OVER** Comprehensive Documentation

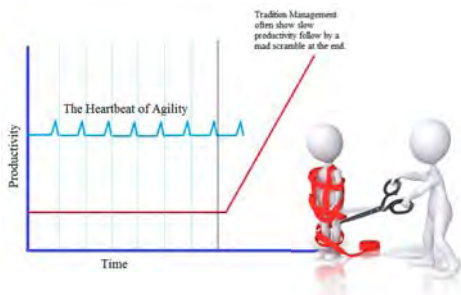
v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Overview The 12 Principles of Agile Software

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity—the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

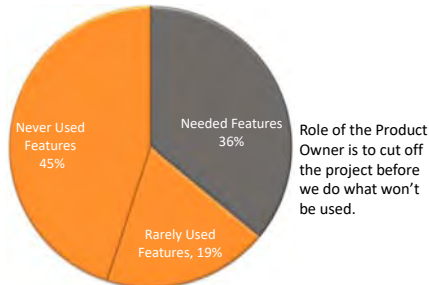
v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

The Heartbeat of Agility



v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Must, Wants & Needs



v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Scrum Basics

Key Terms

- **Pairing** – Two developers working together at one workstation. One writes the code while the other reviews each line as they go.
- **Swarming** – The entire team swarms around a single feature or story e.g. everyone works together on a single story or feature at the same time. It sets a WIP limit of 1.
- **Mobbing** – Combines the concepts of pairing and swarming. The entire team works off of a single keyboard on a single feature.
- **Mini-waterfalls** – team addresses all of the requirements in a given period, then attempts to do all of the design, then moves onto do all of the execution.
- **Fishbowl window** – Long-lived video conference link that allows non-collocated team members to watch each other throughout the day to reduce collaboration lag.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Scrum Basics

- Based on industrial process theory
 - Self-organization
 - Emergence
- Defined Process Control vs. Empirical Process Control
 - **Defined Processes** - Repeatable processes such as in manufacturing. Leads to commoditization. In projects often leads to rework.
 - **Empirical Processes** - Complex processes where it is difficult to have consistent processes. Focuses on 3 keys.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Scrum Basics Empirical Process Keys

- **Visibility** – The aspects of the process that affect the outcome must be visible to those controlling the process & what is seen must be true.
- **Inspection** – The various aspects of the process must be examined frequently enough that unacceptable variances in the process can be detected.
- **Adaptation** – If one or more of the processes are determined out of control the processes change.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Scrum Basics The Scrum Roles

- **Product Owner** – Responsible for representing interests of all stakeholders, obtaining funding, defining initial requirements, ROI, and objectives (Product Backlog).
- **The Development Team** – Develops the functionality. Is self-managing, self-organizing, and cross functional.
- **Scrum Master** – Responsible for the Scrum process, teaching Scrum to everyone, implementing Scrum so it fits with culture, and that everyone follows Scrum rules & practices.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Team Members

- **C** – Committed
- **F** – Focused
- **O** – Open
- **R** – Respected & Respectful
- **C** – Courageous
- **E** – Extreme



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Key Scrum Artifacts A Product / Project Vision

- Serves as a Charter or 5 Line
- Required to begin any project.
- Never longer than a single page.
- Includes 5 key pieces of information: Need, justification, success criteria, prioritization, constraints & assumptions.



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Key Scrum Artifacts

Product Backlog

- A prioritized list of items to be delivered.
- Each item is “relatively independent” of the others.
- Backlog may be reprioritized at any time.
- Items = User Stories or PBIs
- PBI = Product Backlog Item



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Key Scrum Artifacts

Product Backlog

- New requirements are prioritized by your project stakeholders and added to the stack in the appropriate place.
- Fundamentally a single person needs to be the final authority when it comes to requirement prioritization (PO).
- The PO is responsible for representing all other stakeholders.
- The backlog is initially filled as the result of requirements envisioning efforts at the beginning of the project called "populating the backlog".
- Each iteration the team pulls an iteration's worth of work off the top of the stack and commits to implementing it by the end of the iteration.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

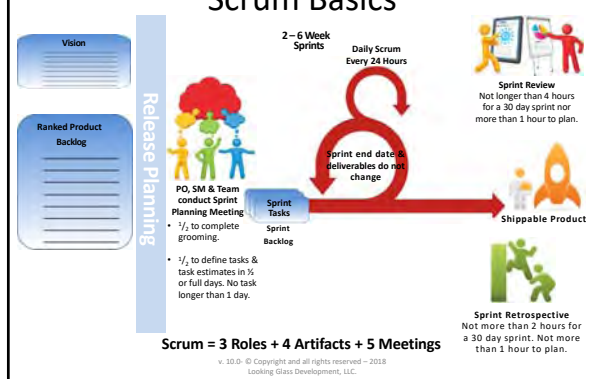
Key Scrum Artifacts

Product Backlog

- Project stakeholders have the right to define new requirements, change their minds about existing requirements, and even reprioritize requirements as they see fit.
- Stakeholders are responsible for making decisions and providing information in a timely manner. On some projects a business analyst, often in the role of product owner, may take on this responsibility.
- The priorities of non-requirement work items are either negotiated by the team with stakeholders or are addressed as part of slack time within the schedule.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Scrum Basics



The Daily Sprint Schedule

	Day 1	Days 2 – X	Last Day
AM	Sprint Planning	Daily Scrum Work & Backlog Grooming	Sprint Review & Sprint Retrospective
PM	Work	Work & Backlog Grooming	1. Free Time 2. Google 20% Time 3. Personal Time

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

The Basics of PM

Sprint Planning Meeting

- PO speak to the sprint goal.
- Acceptance criteria.
- Team forecasts number of stories.
- Team breaks PBIs into tasks.
- Estimate tasks: 0.5 / 1.
- Build sprint burndown and task board.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

The Basics of PM

The Daily Scrum

- 10 to 15 minute meeting for team to answer 3 questions.
- Stand up means STAND UP!
- Target 10 minutes, 15 max.
- Same time every day & don't miss a day.
- Stand in front of the visual progress artifact.
- Everybody is present.
- No typing during the meeting.
- Concentrate on the 2nd and 3rd questions.
- Don't talk to the Scrum Master. Talk to the team.
- Don't solve problems as they become apparent.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

The Basics of PM

Iteration-Based vs Flow-Based Agile

- Iteration-based Agile questions
 - What did I complete yesterday?
 - What am I going to complete today?
 - What impediments do I have?
- Flow-based Agile questions:
 - What do we need to do to advance this piece of work?
 - Is anyone working on anything that is not on the board?
 - What do we need to finish as a team?
 - Are there any bottlenecks or blockers to the flow of work?

v. 10.0 © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

The Basics of PM

Sprint Review

- Summary of sprint by PO.
- PO demonstrates every acceptance criteria of every story delivered.
- Gather feedback from stakeholders and incorporate into product backlog.
- Update release plan and discuss next step.

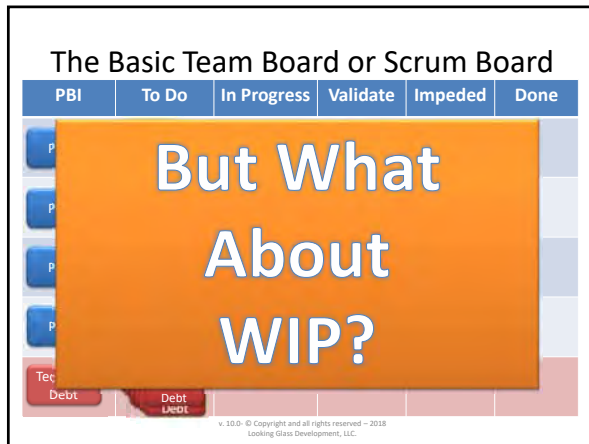
v. 10.0 © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

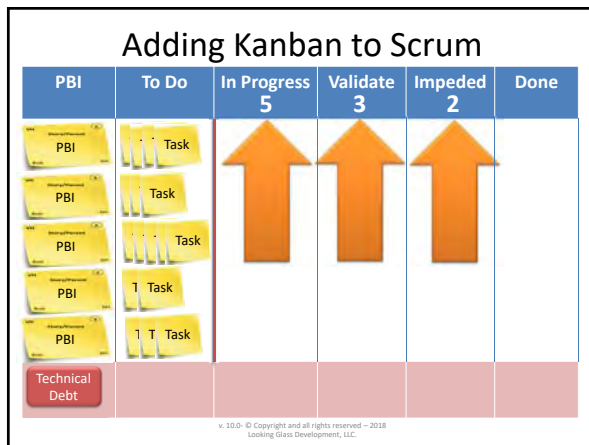
The Basics of PM

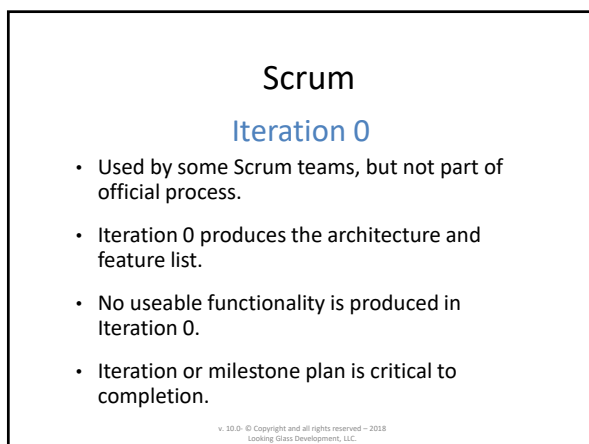
Sprint Retrospective

- The most important Scrum meeting.
- Important to change something in every sprint.
- Remember: "It's not a lesson learned until you do something about."
- Work on only the two most important lessons.

v. 10.0 © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.







Extreme Programming Overview

- XP is a methodology that introduces checkpoints when new requirements can be adopted to improve productivity.
- Iterations of 1 to 2 weeks in length.
- Created by Kent Beck @ Chrysler with Ward Cunningham & Ron Jeffries.
- 12 practices grouped into four (4) areas.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Extreme Programming Basics

- XP's Focus is on:
 - Goals
 - Activities
 - Values
 - Principles
 - Practices



v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Extreme Programming Basics

- **Goals** – Produce high quality software. XP does this through short development cycles.
- **Activities**
 - **Listening** - Programmers must listen to customers & understand the business processes.
 - **Designing** – Software must be designed as components without complexity or dependencies between components.
 - **Coding** – Is the meat of the methodology. It is the most important part according to XP advocates.
 - **Testing**

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Extreme Programming

Core Values

- **Simplicity** – Reduce complexity, extra features & waste. “Find the simplest thing that could possibly work.”
- **Communication** – All the team members know what is expected of them & what others are working on.
- **Feedback** – The team must get impressions & suitability early. It’s about “failing fast”.
- **Courage** – The team has to be willing to put its work out there for others to see. Use pair programming & share code.
- **Respect** – The team is accountable to each other for results.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Extreme Programming

Basics

- Principles
 - **Assume Simplicity** – This is about treating every problem as if its solution were extremely simple.
 - **Embrace Change** – Do not work against change, embrace it.
- Practices (4 areas)
 - Fine Scale Feedback
 - Continuous Process
 - Shared Understanding
 - Programmer Welfare

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Extreme Programming

Core Practices - Fine Scale Feedback

- **Paired Programming** – Two programmers work together at one workstation. One programmer writes while the other reviews & thinks strategically. Pairs are not fixed & change often.
- **The Planning Game** – XP’s main planning process. Occurs once per iteration (typically once per week) Includes two parts: Release Planning & Iteration Planning.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

XP Core Practices

Fine Scale Feedback- Release Planning

- Focused on determining which requirements are included in which near-term release & when they should be delivered.
- Customer & developer are part of this.
- Includes three (3) phases:
 - **Exploration Phase** – Customer provides a short list of high-value requirements for the system that are written as User Stories.
 - **Commitment Phase** – Business & developers commit to the functionality to include & next release date.
 - **Steering Phase** – Plan can be adjusted, new requirements added, existing requirements changed or removed.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

XP Core Practices

Fine Scale Feedback - Iteration Planning

- Focused on planning the tasks & activities for the developers.
- Customer is NOT part of this.
- Includes three (3) phases:
 - **Exploration Phase** – The requirements are translated to specific tasks & recorded on task cards.
 - **Commitment Phase** – The tasks are assigned to programmers & the task durations estimated.
 - **Steering Phase** – Tasks are performed & the results then matched with the original User Story.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

XP Core Practices

Core Practices - Fine Scale Feedback

- **Test Driven Development** – Tests are written before code is written. Tests are automated.
 - Write unit tests.
 - Fail tests. Programmers verify the tests fail.
 - Write code. Programmers write minimal amount of code to pass tests.
 - Pass tests. Code is tested to ensure passage.
 - Refactor. Remove any code smells from the code.
- **Whole Team** – The customer does not always pay the bill, but always uses the system. Customer must be on hand at all times to answer questions.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Extreme Programming

XP Core Practices – Continuous Process

- **Continuous Integration** – XP employs continuous integration & requires the use of a repository loading it every few hours. Integration tests are run automatically.
- **Design Improvement** – Only code what is needed today. When problems occur refactor code to make it simpler & more generic.
- **Small Releases** – Frequent small releases to test are encouraged. Quality is maintained through continuous integration.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Extreme Programming

Core Practices – Shared Understanding

- **Coding Standards** – An agreed upon set of rules that the entire team follows for a consistent style & format. XP advocates self-documenting code that reduces the need for code comments.
- **Collective Code Ownership** – Any pair of developers can make changes to any code & everyone is responsible for all code.
- **Simple Design** – “Simplest” is best approach to software design.
- **System Metaphor** – It is a story that everyone can understand about how the system works. It creates a naming convention that allows customers to guess what class/methods do.

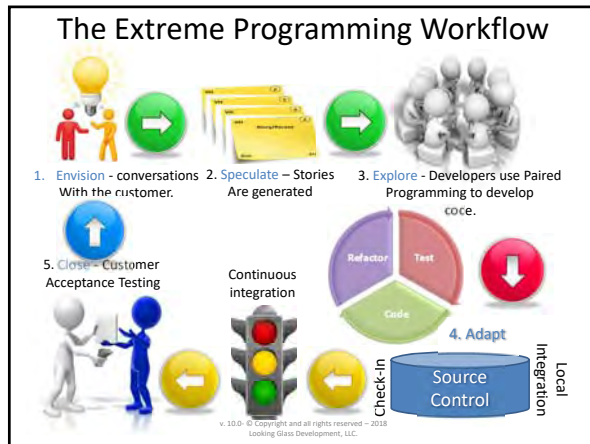
v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Extreme Programming

Core Practices - Programmer Welfare

- **Sustainable Pace** – Programmers should not work more than 40 hours per week & no two weeks in a row should have overtime.
- A key enabler of this concept is frequent code merges of code that are always executable. This code must constantly be tested to ensure it is of a high quality.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.



Extreme Programming

The Basic Steps

- **Envision** – determine the product vision project community, and how the team will work together.
- **Speculate** – develop a feature-based release, milestone, and iteration plan to deliver on the vision.
- **Explore** – deliver tested features in a short timeframe, constantly seeking to reduce the risk and uncertainty of the project.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Extreme Programming

The Basic Steps

- **Adapt** – review the delivered results, the current situation, and the team's performance, and adapt as necessary.
- **Close** – conclude the project, pass along key learnings and celebrate

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

XP vs. Scrum

Scrum

- Teams work in sprints of 2 to 6 weeks.
- Teams do NOT allow changes into their sprints.
- Scrum teams control the order of work, but are informed by the PO.
- Scrum doesn't prescribe any engineering practices

Extreme Programming

- Teams work in iterations of 1 to 2 weeks.
- So long as work has not started on a feature it may be changed or replaced.
- XP teams work on a strict priority order as defined by the customer.
- XP prescribes TDD, automated testing, pair programming, simple design

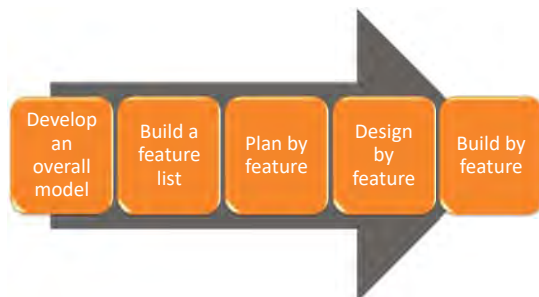
v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Feature-Driven Development (FDD)

- Created by
 - Peter Coad – A leader in the Object Oriented Programming Movement.
 - Jeff De Luca – Co-wrote book with P. Coad. IBM Project Manager. Then founder of Nebulon Corp. in Australia.
- Reference books
 - Java Modeling Color with UML (*Last Chapter*) 1999
 - A Practical Guide to Feature-Driven Development 2002
- Origins
 - Singapore Bank where Peter & Jeff worked together.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Feature-Driven Development (FDD)



v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Feature-Driven Development (FDD)

- **1. Develop Overall Model** – The team holds a kickoff and walks through the scope of the system including all of its context.
 - Detailed domain models are created for each modeling area by small teams and presented for peer review.
 - One of the proposed models is selected to become area domain model.
 - Domain area models are progressively merged into an overall model.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Feature-Driven Development (FDD)

- **2. Build Feature List** – Knowledge from initial modeling is used to identify the list of features by functionally decomposing the domain into subject areas.
 - Subject areas each contain business activities.
 - Steps within each business activity form the basis for a categorized feature list.
 - Features in this respect are small pieces of client-valued functions expressed in the form “<action><result><object>”
 - Features should not take more than two weeks to complete.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Feature-Driven Development (FDD)

- **3. Plan by Feature** – In this step, a development plan is created where ownership of features, or feature sets are assigned as classes to specific programmers.
- **4. Design by Feature** – The Chief Programmer selects a small group of features to be developed in the next two weeks.
 - A Design Package is produced for each feature.
 - Detailed Sequence Diagrams for each feature are produced.
 - Class & Method Prologues are written.
 - The Design Inspection is held.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Feature-Driven Development (FDD)

- **5. Build by Feature**
 - The class owners develop the code for their classes.
 - Unit tests are conducted on the code.
 - Code inspection is conducted.
 - The feature is promoted to the main build.
- Return to step 3 and repeat the process.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Feature-Driven Development (FDD)

- **Domain Object Modeling** – The team explores & explains the domain of the problem.
- **Developing by Feature** – Break the functions down into two-week or shorter chunks & calling them features.
- **Individual Class Ownership** – Areas of code have a single owner (different from XP).
- **Feature Teams** – Small dynamically formed teams that vet designs & allow multiple designs to be evaluated.
- **Inspections** – Reviews that help ensure good-quality design & code.
- **Configuration Management** – Labeling code, tracking changes, & managing source code.
- **Regular Builds** – Make sure new code integrates with existing.
- **Visibility of Progress & Results** – Track progress based on completed work.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Feature Driven Development (FDD)

Scrum

- No formal process.
- No model requirements.
- Product Backlog
- Release plan provides initial view of when PBIs delivered.
- Entire process is iterative

FDD

- Five specific steps.
- Creates overall model 1st thing that is not working software.
- Feature List
- Plan by feature
- Only the last three (3) steps are iterative.

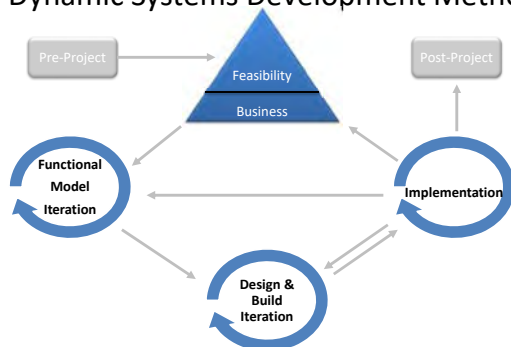
v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Dynamic Systems Development (DSDM)

- Created in 1994 in order to add discipline to RAD.
- Designed to be compatible with ISO 9000 & Prince2.
- It is iterative and incremental.
- Fixes cost, quality and time at project outset.
- In 2007, rebranded DSDM Atern.
- In 2014, went back to just DSDM.
- Often used to provide overall project framework in conjunction with Scrum.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Dynamic Systems Development Method



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Dynamic Systems Development Method

Prerequisites for Using DSDM

- Acceptance of the Atern philosophy before starting work.
- Appropriate empowerment of the Solution Development Team.
- Commitment of senior business management to provide the necessary Business Ambassador involvement.
- Incremental delivery.
- Access by the Solution Development Team to business roles.
- Solution Development Team stability.
- Solution Development Team skills.
- Solution Development Team size.
- A supportive commercial relationship.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

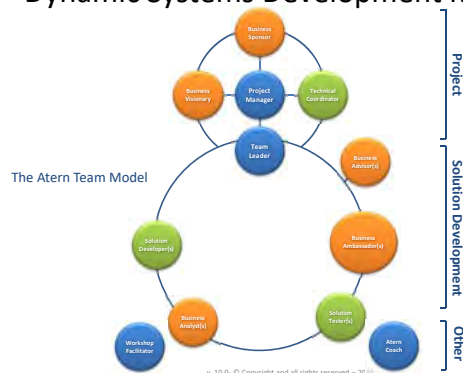
DSDM – Atern Philosophy



Any product must be aligned to clearly defined strategic goals & focus upon early delivery of real benefits to the business. This is best achieved when key stakeholders understand the business objectives, are empowered to an appropriate level & collaborate in order to deliver the right solution. The solution will be delivered in the agreed timescale, according to the priorities set by the business. The stakeholders must be prepared to deliver a fit-for-purpose solution. They must also be prepared to accept that change is inevitable as they understand more about the solution being developed.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Dynamic Systems Development Method



v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Dynamic Systems Development Method

DSDM Eight Principles

- Focus on the business need.
- Deliver on time.
- Collaborate.
- Never compromise quality.
- Build incrementally from firm foundations.
- Develop iteratively.
- Communicate continuously & clearly.
- Demonstrate control.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Dynamic Systems Development Method

1. Focus on the Business Need

- The main acceptance criteria is delivering something that addresses the defined business need.
 - Clearly define the scope of the system.
 - Understand the true business priorities.
 - Establish a sound business case.
 - Seek continuous business sponsorship & commitment.
 - Guarantee the minimum usable subset of features.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Dynamic Systems Development Method

2. Deliver on Time

- Timebox the work.
- Focus on business priorities.
- Always meet deadlines.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Dynamic Systems Development Method

3. Collaborate

- Decisions must be made with users and developers working together quickly.
 - Involve the right stakeholders, at the right time, throughout the project.
 - Ensure that the members of the team are empowered to make decisions on behalf of those they represent without waiting for higher-level approval.
 - Actively involve the business representatives.
 - Build a one-team culture.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Dynamic Systems Development Method

4. Never Compromise Quality

- Set the level of quality at the beginning.
- Ensure that quality does NOT become a variable.
- Design, document, & test appropriately.
- Build in quality by constant review.
- Test early & continuously.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Dynamic Systems Development Method

5. Build Incrementally from Firm Foundations

- Strive for early delivery of business benefit where possible.
- Continually confirm the correct solution is being built.
- Formally re-assess priorities & ongoing project viability with each delivered increment.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Dynamic Systems Development Method

6. Develop Iteratively

- Focus on frequent delivery of products, with assumption that to deliver something “good enough” earlier is always better than to deliver everything “perfectly” in the end.
 - Do enough design up front to create strong foundations.
 - Take an iterative approach to building all products.
 - Build customer feedback into each iteration to converge on an effective business solution.
 - Accept that most detail emerges later rather than sooner.
 - Embrace change – the right solution will not evolve without it.
 - Be creative, experiment, learn, evolve.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Dynamic Systems Development Method

7. Communicate Continuously & Clearly

- Communication & Cooperation among all project stakeholders is required to be efficient & effective.
 - Run daily team stand-up sessions.
 - Use facilitated workshops.
 - Use rich communication techniques such as modelling & prototyping.
 - Present iterations of the evolving solution early & often.
 - Keep documentation lean & timely.
 - Manage stakeholder expectations throughout the project.
 - Encourage informal, face-to-face communication at all levels.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Dynamic Systems Development Method

8. Demonstrate Control

- Use an appropriate level of formality for tracking & reporting.
- Make plans & progress visible to all.
- Measure progress through focus on delivery of products rather than completed activities.
- Manage proactively.
- Evaluate continuing project viability based on the business objectives.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Crystal Overview

- An entire family of methodologies.
- Developed by Alistair Cockburn in mid-1990s.
- Based on observations of successful teams that did not follow formal methodologies.
- Avoids strict or rigid processes found in other methodologies.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Crystal

Cockburn Differentiated Between...

- Methodologies – Sets of elements (e.g. practices or tools).
- Techniques – Skill areas (e.g. developing Use Cases).
- Policies – Dictate organizational musts.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Crystal

Crystal Methods Focus on...

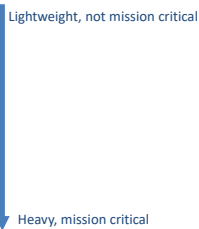
- People
- Interaction
- Community
- Skills
- Talents
- Communications



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Crystal

- Versions scale according to team size.
- The color denotes the weight:
- Crystal Clear - 2-6 people
- Crystal Yellow – 7 – 20 people
- Crystal Orange – 10 – 40 people
- Crystal Orange Web
- Crystal Red - up to 80 people
- Crystal Maroon
- Crystal Diamond
- Crystal Sapphire



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Crystal

Common Crystal 7 Properties

- Frequent delivery
- Reflective improvement
- Close or osmotic communication
- Personal safety
- Focus
- Easy access to expert users
- Automated tests, configuration management, frequent integration

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Crystal

Frequent Delivery

- Software released in iterations weekly or quarterly.
- Possible to have multiple iterations per release.
- Problems found and fixed early on.
- Customers get to ensure the project is going the way they want it to go.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Crystal

Reflective Improvement

- Developers dedicate time to improving the development process.
- Reflection workshops are held every few weeks to help find processes that are working & which ones need to be modified.
- Iteration helps determine if a process is working or not.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Crystal

Close or Osmotic Communication

- Development teams **MUST** be in the same room.
 - Developers do not need to break concentration to move somewhere else.
 - Information flows quickly through the team.
 - Communication overhead is reduced.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Crystal

Personal Safety

- Team members must be able to speak freely in a group without being ridiculed.

Focus

- Focus on a task long enough for progress to be made.
 - 2 hour periods where the developers have no interruptions.
 - Developer assigned to a project for at least 2 complete days.
- Clear definition and goals of the project.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Crystal

Easy Access to Expert Users

- Developers work with experts in the field of the project who will also be end-users.
- Expert will answer questions and suggest solutions or improvements.
- Minimum: meet once a week for 2 hours and be reachable by phone.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Crystal

Automated Tests, Configuration Management Frequent Integration

- Spot errors and problems that arise from changes being made.
- Done regularly
 - Problems spotted early on.
 - Problems are less likely to grow.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

7. See the Whole
Software systems are a product of their interactions and not just a sum of parts.
 • Defects accumulate during the development process.
 • By decomposing big tasks into smaller ones and standardizing the stages of development defect root causes can be found and eliminated.
 • Think big, act small, fail fast, learn rapidly.

6. Build Quality In
The customer needs to have an overall perceived integrity of the system. This is being realized, delivered, deployed, tested, how intuitive is it to use, what is price, and how well does it solve problems.
 Conceptual integrity means the system's disparate components work well together as a whole with balance between flexibility, maintainability, efficiency and responsiveness.
 • Refactoring is about keeping simplicity, clarity, minimum amounts of features in the code.
 • Integrity is verified through testing to ensure the system does what the customer expects.

1. Eliminate Waste
You must first recognize waste before you can eliminate it.
 • The fastest, most efficient way to incorporate into the development process is to build quality into the software from the start.
 • The fastest, most efficient way to incorporate into the development process is to build quality into the software from the start.

2. Encourage progress, catch mistakes, removing impediments.
 • Discourage micro management.
 • People are more than just resources. They require motivation and a high purpose.
 • The developer should be given autonomy to solve the customer's problem.

Lean Software Development (LSD)

Key Tools & Concepts

- Translation of Lean Manufacturing to IT.
- Originated by Mary & Tom Poppendieck.
- **Pull Systems** - Kanban
- **Queueing Theory** - The mathematical study of waiting lines, or queues. In queueing theory a model is constructed so that queue lengths and waiting time can be predicted.
- Value Stream Mapping
- Set-Based Development
- Seeing Waste
- Motivation
- Measurements

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Lean Software Development (LSD) Seven Wastes (TIMWOOD)

- **Transportation** – Each time a product is moved it stands the risk of being damaged, lost, or delayed.
- **Inventory** – Raw materials, WIP, or finished goods represent capital outlays that have not yet produced an income.
- **Motion** – Refers to the damage that the production process inflicts on the entity that creates the product.
- **Waiting** – Whenever goods are not in transport or being processed, they are waiting.
- **Over-Processing** – Occurs any time more work is done on a piece than is required by the customer.
- **Over-Production** – Occurs when more product is produced than is required at a time by the customer.
- **Defects** – Whenever they occur, extra costs are incurred reworking the part, rescheduling production, etc.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Kanban

- Means signboard in Japanese.
- It is a scheduling system used in Lean Production to improve the JIT flow.
- Kanban limits WIP by defining the maximum number of stories to be worked at a time.
- Typically, it uses a series of columns showing value being added to work flowing left to right.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Kanban

- Best used when a team or organization is in need of:
 - **Flexibility** – The team is **NOT** bound by timeboxed and will work on the highest priority item in the backlog.
 - **Focus on continuous delivery** – Teams are focused on flowing work through the system to completion and **NOT** beginning new work until WIP is completed.
 - **Increased productivity and quality** – Both factors are improved by limited WIP.
 - **Increased efficiency** – Checking each task for value adding or non-value added activities and remove all non-value added activities.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Kanban

- Best used when a team or organization is in need of:
 - **Team member focus** – Limiting WIP allows the team to focus on the current work.
 - **Variability in workload** – When there is unpredictability in the way work arrives teams cannot make predictable commitments therefore it becomes critical to manage WIP.
 - **Reduction in waste** – Transparency makes waste visible so it can be removed.

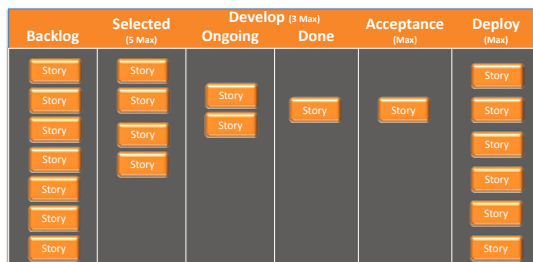
v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Kanban

- It's a pull system to take work when capacity is available instead of pushing work.
- Taiichi Ohno developed Kanban at Toyota in 1953.
- Best book is KANBAN – Successful Evolutionary Change for Your Technology Business by David J. Andersson.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Task/Kanban Board



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Kanban

Five Core Principles of Kanban

- **Visualize the Workflow** – Have some way to visualize the workflow.
- **Limit WIP** – Keep the amount of WIP low.
- **Manage Flow** – Track the flow through the system.
- **Make Process Policies Explicit**
- **Improve Collaboratively**

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Scrum vs. Kanban

Scrum

- Scrum Boards reset before each sprint.
- Scrum Boards typically have To-Do, In Progress, Validate, Impeded & Done.
- Scrum Boards change infrequently.

Kanban

- Kanban doesn't use timeboxed iterations (sprints) so it uses a continuous flow of work with new work added when there is capacity.
- Kanban Boards tend to have more columns.
- Kanban Boards change frequently.

Teams often start with Scrum and add Kanban which leads to them not doing Scrum "by the book" any longer.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Test Your Assumption

- You must know which methodology is best for which situation:
 - **Scrum** as a basic Agile starting point.
 - **XP** is ruthless in testing.
 - **FDD** offers robust & specific modeling techniques.
 - **DSDM** offers suitability filters for how well a process fits a project.
 - **Crystal** works on projects of different sizes & complexities.
 - **Kanban** for modification & adaptation.
 - **Cockburn's Shu-Ha-Ri Model**
 - **Shu** - Obey the rules.
 - **Ha** - Consciously move away from the rules.
 - **Ri** - Unconsciously find your own path.
- Critical to align team size & methodology.
- Project visibility.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Scaling Agile

- Most Agile life cycles are designed for small teams and projects.
- Several techniques are available to scale Agile:
 - Scaled Agile Framework (SAFe)
 - Nexus
 - Large Scale Scrum (LeSS)
 - Disciplined Agile Delivery (DAD)

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

SAFe 3.0

- Interactive knowledge base for implementing agile practices at an enterprise scale.
- **NOT** for corporate IT environments, is designed for **BIG** software products.
- 3 levels for SAFe: Team, Program, Portfolio.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

SAFe 3.0 SAFe Core Values

- **Alignment** – to a common mission using 3 tools:
 - **Release Planning** – Each PI begins with a Release Planning Meeting with all teams together.
 - **Scrum of Scrums** – RTE & Scrum Masters meet twice weekly in front of the Program Board to discuss progress, risks & dependencies.
 - **System Demo** – Assisted by Systems Team @ end of every iteration to show working solutions.
- Visibility
- Program Execution
- Code Quality

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

SAFe 3.0

Level I - Portfolio

- Strategic themes connect portfolio or organizational strategic objectives.
- Budgeting for each ART or Program done at this level.
- Epics are created to fund cross ART training or deliverables.
- Use Kanban system at this level to represent ARTs.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

SAFe 3.0

Level II - Program

- 50 – 125 people called an Agile Release Train (ART) is the specific program. If team misses one train they can catch the next one.
- Program Increment (PI) are the timeboxed increments.
- Each PI is 5 iterations by default. 4 are focused on the backlog, 1 is IP Iteration for Planning. This is used to deal with the unexpected or creative. Run Inspect & Adapt Workshop where the team works to improve process.
- Has a separate Program Backlog that represents larger features that are broken down to create the team backlog. Owned by Product Manager.
- Guided by a Release Train Engineer (RTE) is program manager.
- Train System Architect facilitates process of developing infrastructure for future PIs.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

SAFe 3.0

Level III Team

- Uses Kanban, Scrum or Scrumban.
- 2 week iterations.
- Not every sprint produces a potentially shippable increment.
- Changes some terms which Schwaber & others don't like.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Nexus

What is Nexus?

- **Nexus** – Is a unit of development in Scaled Professional Scrum.
- **Nexus** – Is a framework consisting of roles, events artifacts, and techniques that bind the work of 3 to 9 Scrum teams.
- **Nexus** – Is an exoskeleton that rests on top of multiple Scrum Teams to create an Integrated Increment.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Nexus

Nexus Consists of...

- Roles are the same as Scrum.
- Adds one new role the Nexus Integration Team:
 - Product Owner(s)
 - Scrum Master(s)
 - Nexus Integration Team Members

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Nexus

Nexus Consists of...

- All Scrum teams use the same, single product backlog.
- Nexus events:

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Nexus

Nexus Process Flow

- Refine the product backlog.
- Nexus Sprint planning
 - Backlog items reviewed
 - PBIs assigned to teams & they create their Sprint backlog.
 - Each Scrum team plans its own sprint.
 - Teams align sprint goal(s) to Nexus goal(s).
- Team develops features.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Nexus

Nexus Process Flow

- Nexus Daily Scrum:
 - Was the previous day's work successfully integrated? If not, why?
 - What new dependencies have identified?
 - What information needs to be shared across teams in the Nexus.
 - Each team then holds its own Daily Scrum.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Nexus

Nexus Process Flow

- Nexus Sprint review – All teams meet with Product Owner to review the integrated increment. Adjustments may be made to the backlog.
- Nexus Sprint retrospective
 - Shared challenges are discussed.
 - Each team holds its own retrospective.
 - Teams meet & agree on how to visualize & track the identified actions.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Nexus

Each Retrospective Should...

- Was any work left undone? Did the Nexus generate technical debt?
- Were all artifacts, particularly code, frequently (as often as every day) successfully integrated?
- Was the software successfully built, tested, & deployed often enough to prevent the overwhelming accumulation of unresolved dependencies?

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Nexus

Refinement Meetings

- The greater the complexity & dependencies, the more the Product Backlog items must be refined to remove dependencies.
- Helps to forecast which team will deliver each item.
- Identifies dependencies across teams.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

LeSS

Overview

- Large-scale Scrum is regular Scrum applied to large-scale development.
- Created by Craig Larman and Bas Vodde.
- There are two frameworks depending on project size:
 - Framework 1 – Projects up to 10 teams.
 - Framework 2 – Larger projects.
- Is an organizational design framework.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

LeSS

LeSS / Scrum Commonalities

- A single Product Backlog (because it's for a product, not a team).
- One Definition of Done for all teams.
- One Potentially Shippable Product Increment at the end of each Sprint.
- One Product Owner.
- Many complete, cross-functional teams (with no single-specialist teams).
- One Sprint.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

LeSS

LeSS / Scrum Differences

- **Sprint Planning Part 1:** In addition to the one PO, it includes people from all teams. Let team members self-manage to decide their division of Product Backlog Items. Team members also discuss opportunities to find shared work and cooperate.
- **Sprint Planning Part 2:** Held independently (and usually in parallel) by each Team, though sometimes for simple coordination and learning two or more Teams may hold it in the same room.
- **Daily Scrum:** This is also held independently by each Team, though a member of Team A may observe Team B's Daily Scrum, to increase information sharing.
- **Overall PBR:** There may be an optional and short overall Product Backlog Refinement (PBR) meeting that includes the one PO and people from all teams. Purpose is to decide which teams are likely to implement which items and therefore select those items for later in-depth single-team PBR. It is also a chance to increase alignment with the Product Owner and all teams.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

LeSS

LeSS / Scrum Differences

- **Product Backlog Refinement:** The only requirement in LeSS is single-team PBR, the same as in one-team Scrum. But a common and useful variation is multi-team PBR, where two or more Teams are in the same room together, to increase learning and coordination.
- **Sprint Review:** In addition to the one Product Owner, it includes people from all teams, and relevant customers/users and other stakeholders. For the phase of inspecting the product increment and new items, consider a "bazaar" or "science fair" style: a large room with multiple areas, each staffed by team members, where the items developed by teams are shown and discussed.
- **Overall Retrospective:** This is a new meeting not found in one-team Scrum, and its purpose is to explore improving the overall system, rather than focusing on one Team. The maximum duration is 45 minutes per week of Sprint. It includes the Product Owner, Scrum Masters, and rotating representatives from each Team.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Disciplined Agile Development

Overview

- A people-first, learning-oriented, hybrid agile approach to IT solution delivery.
- Makes use of Scrum, Agile Modeling, XP, UP, Kanban, Lean Software Development, Outside In Development & others.
- Looks at full, end-to-end delivery life cycle from project initiation to solution delivery.
- Provides technical practice advice like XP as well as modeling, documentation, & governance.
- Less prescriptive than Scrum by being more goal-driven. Provides significant advice & alternatives.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Disciplined Agile Development

Overview

- Draws from UP.
- Projects divided into three phases:
 - Inception
 - Construction
 - Transition
- Scott Ambler creator

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Disciplined Agile Development

The Backlog

- **Go beyond functional requirements.** Teams often must complete non-requirement related work such as take training, review products of other teams, address defects. These need to be on the backlog.
- **Take a risk-value approach.** Common risks include the need to come to stakeholder consensus early in the project, or the need to prove that your architecture strategy, actually works. DAD teams look at their work item stack early in the project, typically during the Inception or iteration 0 to identify requirements which exhibit these technically risky features.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Disciplined Agile Development

The Backlog

- **Model a bit ahead.** What if a work item is very complex, requiring a bit more thinking that what generally occurs in an iteration planning session? DAD teams adopt the Look-Ahead Modeling practice and look ahead an iteration or two and invest the time to explore complex upcoming items to reduce the overall project risk. Modeling a bit ahead is called backlog grooming in Scrum.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Disciplined Agile Development

Lifecycle Versions

- **Agile/Basic** – Based on Scrum, but extended to provide a streamlined strategy from beginning to end.
- **Lean/Advanced** – Based on Kanban.
- **Continuous Delivery** – Stable team based on Kanban and Lean.
- **Exploratory/Lean Startup** – Base on Lean Startup strategies

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Disciplined Agile Development

Roles

Primary Roles:

- Team Lead
- Product Owner
- Architecture Owner
- Team Member
- Stakeholder



Secondary Roles

- Specialist
- Independent Tester
- Domain Expert
- Technical Expert
- Integrator

Primary roles occur on all DAD projects regardless of scale.

Secondary roles only occur at scale & sometimes are only temporary.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

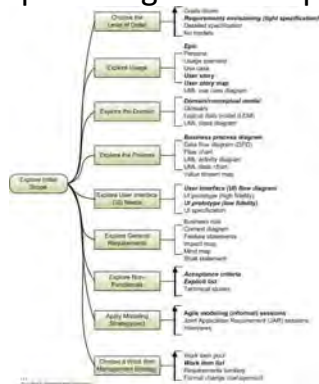
Disciplined Agile Development

DAD vs. Scrum

- DAD places more emphasis on architecture & technical risk reduction.
- Adds the role of Architecture Owner.
- Changes names in Scrum such as Scrum Master becoming Team Lead.
- Is far less prescriptive

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Disciplined Agile Development



PMOs

Project Management Office (PMO) —
Centralizes the management of projects.
Typically 3 structures:

- **Supportive** — Providing a consultative role by providing templates, best practices, training, access to information, & lessons learned.
- **Controlling** — Provide support & require compliance.
- **Directive** — Take control of the projects by directly managing the projects.
- An Agile PMO is value-driven based on a customer collaboration mindset and is invitation-oriented.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

PMOs

Often, Agile PMOs become centers of excellence providing:

- Development and implementation of standards.
- Training, mentoring and organizational learning.
- Multi-project management.
- Stakeholder management
- Recruiting, selecting, and evaluating team leaders.
- Executing specialized tasks for projects.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Organizational Structure Differentiation vs. Integration

The need for specialized areas of expertise (Production, Marketing, Finance, etc.). The level of differentiation hinges on the needs of the organization.



The need for coordinated and cross-functional efforts to accomplish organizational tasks.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Organizational Structure

Many factors influence structure including:

- Geography
- Functionalize structure
- Size of project deliverables
- Allocation of people to projects
- Procurement-heavy organizations

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Organizational Structure

Project Characteristics						
Organizational Structure Type	Matrix Structure	Project Manager's Authority	Project Manager's Role	Resource Availability	Who Manages the Project Budget	Project Management Administration Staff
Organic or Simple	Formal project meeting only by title	Little or none	Part time; may or may not be a designated job title combination	Little or none	Owner or specialist	Little or none
Functional or Specialist	Not being done in a engineering, manufacturing	Little or none	Part time; may or may not be a designated job title combination	Little or none	Functional manager	Part time
Multi-dimensional	One of products, one of projects, portfolio, programs, geographic regions, customer type	Little or none	Part time; may or may not be a designated job title combination	Little or none	Functional manager	Part time
Matrix	Job function	Low	Part time; may or may not be a designated job title combination	Low	Functional manager	Part time
Balance	Job function	Low to moderate	Part time; understood in the business as a part and may not be a designated job title or combination	Low to moderate	Mixed	Part time
Skewed	No job function, with project manager as a function	Moderate to high	Full-time designated job title	Moderate to high	Project manager	Full time
Project-based (simple, balanced, or skewed)	Project	High to almost total	Full-time designated job title	High to almost total	Project manager	Full time
Hybrid	Hybrid structure with roles of products or projects with other people	Low to moderate	Full-time or part time	Low to moderate	Mixed	Could be full-time or part time
Hybrid	One of other types	Mixed	Mixed	Mixed	Mixed	Mixed
Hybrid	One of other types	High to almost total	Full-time designated job title	High to almost total	Project manager	Full time

Value-Driven Delivery

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Domain Tasks – Define Positive Value

1. Define deliverables by identifying units that can be produced incrementally in order to maximize their value to stakeholders while minimizing non-value added work.
2. Refine requirements by gaining consensus on the acceptance criteria for features on a just-in-time basis in order to deliver value.
3. Select & tailor the team's process based on project & organizational characteristics as well as team experience in order to optimize value delivery.

Domain Tasks – Avoid Potential Downsides

4. Plan for small releasable increments by organizing requirements into minimally marketable features/minimally viable products in order to allow for the early recognition & delivery of value.
5. Limit increment size & increase review frequency with appropriate stakeholders in order to identify & respond to risks early on & at minimal cost.
6. Solicit customer & user feedback by reviewing increments often in order to confirm & enhance business value.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Domain Tasks – Prioritization

7. Prioritize the units of work through collaboration with stakeholders in order to optimize the value of the deliverables.
8. Perform frequent review & maintenance of the work results by prioritizing & maintaining internal quality in order to reduce the overall cost of incremental development.
9. Continuously identify & prioritize the environmental, operational, & infrastructure factors in order to improve the quality & value of the deliverables.

Domain Tasks – Incremental Development

10. Conduct operational reviews &/or periodic checkpoints with stakeholders in order to obtain feedback & corrections to the work in progress & planned work.
11. Balance development of deliverable units & risk reduction efforts by incorporating both value producing & risk reducing work into the backlog in order to maximize the total value proposition over time.
12. Re-prioritize requirements periodically in order to reflect changes in the environment & stakeholder needs or preferences in order to maximize the value.
13. Elicit & prioritize relevant non-functional requirements (such as operations & security) by considering the environment in which the solution will be used in order to minimize the probability of failure.
14. Conduct frequent reviews of work products by performing inspections, reviews, &/or testing in order to identify & incorporate improvements into the overall process & product or service.

Looking Glass Development, LLC.

Value-Driven Delivery

- **Value-Driven Delivery** – Focusing on delivering real business value. The most important features first.
- Consider technical dependencies & risks.
- Common mantra in Agile.
- Let's start by defining value-driven delivery. The reason projects are undertaken is to generate business value, be it to produce a benefit or improve a service. Even safety and regulatory compliance projects can be expressed in terms of business value by considering the business risk and impact of not undertaking them. If value then is the reason for doing projects, value driven delivery is the focus of the project throughout the project planning, execution, and control processes.
- It is the big picture view, the wearing of the sponsor's hat when making decisions. By the project manager and team assuming this viewpoint, there is an opportunity to incorporate unique technical insights, such as technical dependencies or risk reduction steps, into the selection of features for a release that the sponsor may not be aware of. However value driven delivery remains a guiding vision for much local decision making, the selecting of choices that maximize the value delivered to the business or customer.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Assessing Value

- Payback Period
- Return on Investment (ROI)
- Future Value
- Present Value
- Net Present Value (NPV)
- Internal Rate of Return (IRR)
- Benefit / Cost ratio (BCR or BCI)
- Focus on when to use these tools



v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Drive Delivery

Assessing Value

Discounting or Present Value – Value today of funds available in the future.

$$PV = FV / (1 + i)^n$$

- If you want \$1,000 in three (3) years how much do you have to invest today at 8% to receive your \$1,000?
- End of Yr. 1 = $\$1,000 / (1 + 8\%) = \925.93
- End of Yr. 2 = $\$925.93 / (1 + 8\%) = \857.34
- End of Yr. 3 = $\$857.34 / (1 + 8\%) = \793.83

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Assessing Value

- **Net Present Value** – Present Value minus Present cost.
- **Internal Rate of Return** - Average rate of return earned over the life of the project. It is where discounted cash flow – up front costs = 0.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Planning Value

- **Chartering**
 - Exists in Agile projects.
 - Focused on what not how.
 - Shorter document, typically one page.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

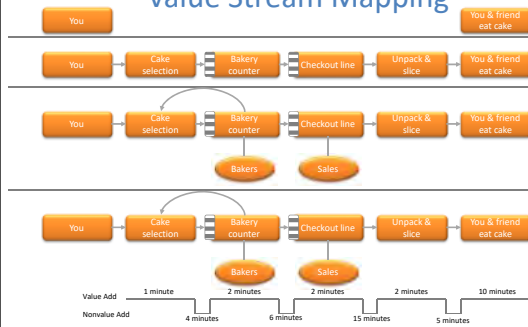
Value Stream Mapping

- Lean Manufacturing technique
- Uses visual maps of process
- 6 Step process:
 1. Identify the product or service that you are analyzing.
 2. Create a value stream map of the current process, identifying steps, queues, delays, & information flows.
 3. Review the map to find delays, waste & constraints.
 4. Create a new value stream map of the desired future state of the process, optimized to remove or reduce delays, waste & constraints.
 5. Develop a roadmap for creating the optimized state.
 6. Plan to revisit the process in the future to continually improve.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Value Stream Mapping



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Value Stream Mapping

- Total Cycle Time = Value Added Time + Non-Value Added Time
- $TCT = (1+2+2+2+10) + (4+6+15+5) = 47 \text{ Minutes}$
- Process Cycle Efficiency = $\frac{\text{Total Value Added Time}}{\text{Total Cycle Time}}$
- $PCE = \frac{17}{47} = 36\%$

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

- Calculate the process cycle efficiency for each row.

NonValue Add	Value Add	Total Time	Efficiency
11	4		
10	6		
8	7		
3	8		
5	9		
7	12		
9	10		
9	5		
4	13		

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

- Calculate the process cycle efficiency for each row.

NonValue Add	Value Add	Total Time	Efficiency
11	4	15	27%
10	6	16	38%
8	7	15	47%
3	8	11	73%
5	9	14	64%
7	12	19	63%
9	10	19	53%
9	5	14	36%
4	13	17	76%

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

- Poppendieck's 7 Lean Wastes Manufacturing to Software.

Waste	Description	Example
Partially Done Work	Work started, but not completed; partially done work can entrophy	<ul style="list-style-type: none"> Code waiting for testing Specs waiting for development
Extra Processes	Extra work that does not add value	<ul style="list-style-type: none"> Unused documentation Unnecessary approvals
Extra Features	Features that are not required, or are thought of as "nice-to-haves"	<ul style="list-style-type: none"> Gold-plating Technology features
Task Switching	Multitasking between several different projects when there are context-switching penalties	<ul style="list-style-type: none"> People on multiple projects
Waiting	Delays waiting for reviews and approvals	<ul style="list-style-type: none"> Waiting for prototype reviews Waiting for document approvals
Motion	The effort required to communicate or move information or deliverables from one group to another. If teams are not co-located, this effort may need to be greater	<ul style="list-style-type: none"> Distributed teams Handoffs
Defects	Defective documents or software that needs correction	<ul style="list-style-type: none"> Requirements defects Software bugs

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

- Customer-valued prioritization
 - Working on the things that yield the greatest return for the customer.
 - Scrum = Product Backlog
 - FDD = Feature list
 - DSDM = Prioritized Requirements List
- MoSCoW Prioritization Schemes
 - Must Have
 - Should Have
 - Could Have
 - Would like to have, but not this time

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

- **Monopoly Money** = to total project budget.
- **100 Point Method** – Used with Use Cases from Leffingwell & Widrig.
- **Kano Analysis** – Customer satisfaction and product development theory developed in the 1984 by Professor Noriaki Kano designed to classify customer preferences into three categories.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

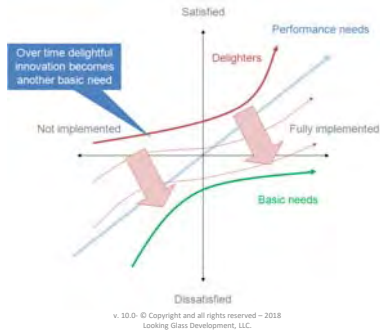
Value-Driven Delivery

Kano Analysis

- **Performance Needs** – These requirements can both satisfy & dissatisfy customers. They are at the top of the customers' mind. Customers will also talk about them readily when asked what is important. You must choose the correct ones of these.
- **Basic Needs** – Having these requirements will NOT result in customer satisfaction, but NOT having them will result in dissatisfaction. Customers don't give these items a thought unless they are absent. They are sometimes referred to as MUST HAVES.
- **Excitement Needs** – Customers are delighted when these are delivered, but their absence doesn't cause dissatisfaction. They are often called UNIQUE SELLING or VALUE PROPOSITIONS.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery Kano Analysis

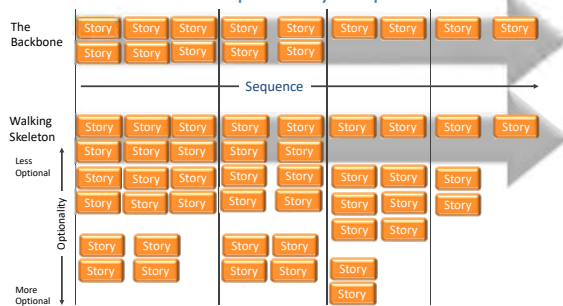


Value-Driven Delivery

- Requirements Prioritization Model
 - By Karl Wiegers
 - Measures benefit, penalty, cost & risk of each feature.
 - Scores from 1 to 9
- **Relative Prioritization** - A simple list removes the categories that people tend to fixate on from the debate and allows the focus of the discussion to be on priorities. It also provides a framework for deciding if and when to incorporate changes. When change is requested, the team can ask the business representatives, "What items are more important than this change?"

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Value-Driven Delivery A Sample Story Map



Value-Driven Delivery

- **Risk** – an uncertain event or condition that, if realized, has a positive or negative impact on at least one project objective (such as time, cost, scope or quality).
- Risks can have one or more causes and one or more impacts.
- Negative risks are anti-value.



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Major Risk Classes

- **Known Risks** - Can be analyzed, possible to plan. Contingency reserve or other plans.
- **Unknown Risks** - Cannot be managed proactively. General contingency or management reserve.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Agile Helps Mitigate Risks

- **Schedule Risks** - In waterfall it can be a long time before a product is ready for release. In Agile this can be shorter, sometimes a few weeks.
- **Budget Risks** – Agile estimates are no more accurate, but it is easier to manage.
- **Cancellation Risk** – Waterfall projects tend to be cancelled late. An Agile project produces functionality quickly so the project can be cancelled.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Agile Helps Mitigate Risks

- **Scope Creep** – In waterfall projects scope is added without any being removed. The backlog forces hard decisions.
- **Requirements Error** – Waterfall projects specify requirements long before they are delivered. An incorrect requirement can generate significant effort before the error is discovered.
- **Technology Risks** – Many projects require unproven technologies. Waterfall extends the time it takes to discover failed tech. Agile allows for rapid discovery.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Agile Helps Mitigate Risks

- **Security Risks** – Waterfall projects often take significant time before system security can be tested. Agile projects test security quickly, often and at regular intervals.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery



v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Expected Monetary Value (EMV)

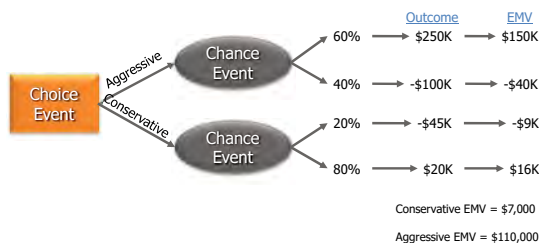
- Calculates the average outcome when future events are uncertain

	Cost	Probability	Product
Optimistic Outcome	\$150,000	.20	\$30,000
Likely Outcome	\$225,000	.50	\$112,500
Pessimistic Outcome	\$300,000	.30	\$90,000
			\$ 232,500

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

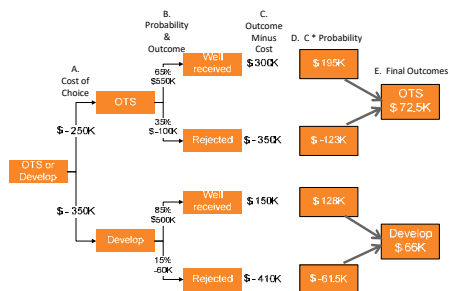
Value-Driven Delivery

Decision Tree Analysis



v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Value-Driven Delivery



v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Value-Driven Delivery

- Use EMV to determine risk value.
- Insert risks into backlog with requirements sorted by cost.

Prioritized Risk List		Prioritized Requirements		Risk-Adjusted Backlog
Risk 1 \$10,000 x 30% = \$6,000		Requirement 1 \$6,500		Requirement 1 \$6,500
Risk 2 \$7,000 x 50% = \$3,500		Requirement 2 \$5,000		Risk 1 \$6,000
Risk 3 \$12,000 x 50% = \$3,000		Requirement 3 \$4,000		Requirement 2 \$5,000
Risk 4 \$5,000 x 20% = \$2,000		Requirement 4 \$3,000		Requirement 3 \$4,000
Risk 5 \$6,000 x 33% = \$1,000		Requirement 5 \$2,500		Risk 2 \$3,500
		Requirement 6 \$1,000		Requirement 4 \$3,000
				Risk 3 \$3,000
				Requirement 5 \$2,500
				Risk 4 \$2,000
				Requirement 6 \$1,000
				Risk 5 \$1,000

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Value-Driven Delivery

- You have been asked to establish an estimated project cost using Expected Monetary Value (EMV). If the project has a best case estimate of U.S. \$10,000 with a probability of 20%, a most likely case estimate of U.S. \$12,000 with a probability of 50%, and a worst case estimate of U.S. \$14,400 with a probability of 30% what is the EMV for the project?
A. U.S. \$12,320
B. U.S. \$12,400
C. U.S. \$13,010
D. U.S. \$13,260
- You have been asked to establish an estimated project cost using Expected Monetary Value (EMV). If the project has a best case estimate of U.S. \$15,000 with a probability of 30%, a most likely case estimate of U.S. \$19,500 with a probability of 50%, and a worst case estimate of U.S. \$26,325 with a probability of 20% what is the EMV for the project?
A. U.S. \$19,190
B. U.S. \$19,515
C. U.S. \$20,110
D. U.S. \$20,350

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Value-Driven Delivery

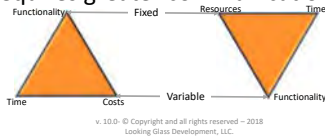
- You have been asked to establish an estimated project cost using Expected Monetary Value (EMV). If the project has a best case estimate of U.S. \$10,000 with a probability of 20%, a most likely case estimate of U.S. \$12,000 with a probability of 50%, and a worst case estimate of U.S. \$14,400 with a probability of 30% what is the EMV for the project?
A. U.S. \$12,320
B. U.S. \$12,400
C. U.S. \$13,010
D. U.S. \$13,260
- You have been asked to establish an estimated project cost using Expected Monetary Value (EMV). If the project has a best case estimate of U.S. \$15,000 with a probability of 30%, a most likely case estimate of U.S. \$19,500 with a probability of 50%, and a worst case estimate of U.S. \$26,325 with a probability of 20% what is the EMV for the project?
A. U.S. \$19,190
B. U.S. \$19,515
C. U.S. \$20,110
D. U.S. \$20,350

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Agile Contracting

- Traditional contracting attempts to fix scope & cost. This often leads to overruns.
- Agile contracting fixes time & costs leaving scope flexible.
- This requires greater communication.



Value-Driven Delivery

Agile Contracting

- **DSDM Contracting** – Focuses on work being “fit for business purpose” & passing tests rather than matching specifications.
- **Jeff Sutherland** – “Money for Nothing & Change for Free” suggests early termination options & flexibility in making changes. Standard fixed price contract + T&M for additional work + a “change for free” clause if they work with the team on every iteration.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Agile Contracting

- **Graduated Fixed Price Contracts**
 - Thorup & Jensen
 - Both parties share risk
 - Hourly rates are defined based on delivery

Completion	Graduated Rate	Total Fee
Early	\$200 / Hour	\$150,000
On Time	\$180 / Hour	\$175,000
Late	\$160 / Hour	\$200,000

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Agile Contracting

- Fixed Price Work Packages –
 - Work is estimated at a lower level.
 - Seller is allowed to re-estimate remaining work packages as project progresses.
 - Customer allowed to focus on greatest value.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Agile Contracting

- **Multi-tiered structure** – Fixed items placed in master agreement. Items subject to change placed in schedule of services.
- **Emphasize value delivered** – Relationship governed by fixed milestones or phase gates based on value-driven deliverables.
- **Fixed price increments** – Decompose the scope into fixed price micro-deliverables such as user stories.
- **Not-to-exceed time and materials** – Limits the overall budget to a fixed amount. Allows customer to exchange requirements.
- **Graduated time and materials** – Supplier is rewarded with a higher hourly rate when delivery is earlier than the contracted deadline. The contractor suffers a rate reduction for late delivery.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Agile Contracting

- **Early cancellation option** – Customer can cancel the contract with only a limited cancellation fee should they desire.
- **Dynamic scope option** – Customer has the option to vary project scope at specified points in the project.
- **Team augmentation** – Suppliers services are embedded directly into the customer's organization. Customer funds the team not a specific scope.
- **Favor full-service suppliers** – To diversify risk, customer uses multiple suppliers where each often focuses on a single area. This often creates engagement and communication risks.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Why not Gantt Charts & other software?

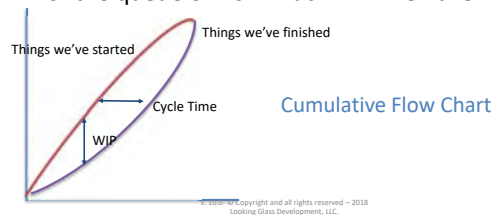
- Data accuracy perception increases.
- Barriers for stakeholder interaction increase.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Little's Law

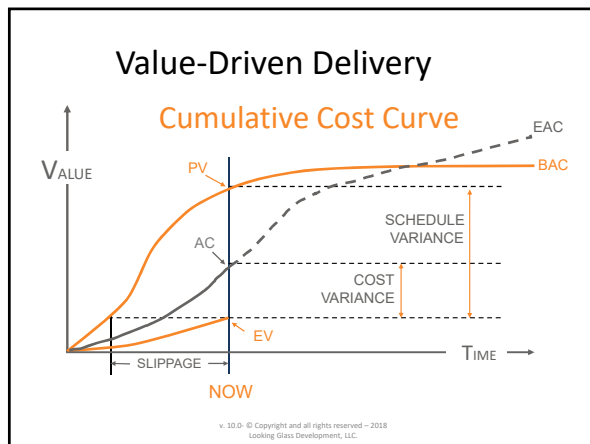
- The cycle time, how long we are going to have to wait for benefits, is proportional to the size of the queue or how much WIP we have.

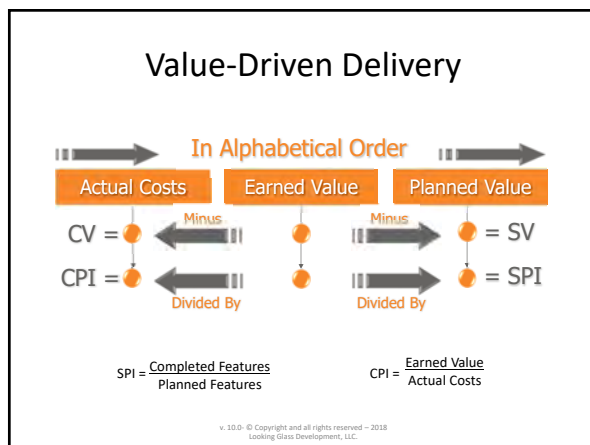


Value-Driven Delivery

- Demonstrations – Critical to confirming success.
 - We learn about the differences between what is asked for and what was interpreted & built.
 - We learn about new or adjusted functionality.
- IKIWISI – I'll Know It When I See It.
- Agile favors empirical and value-based measurements instead of predictive measurements.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.





Value-Driven Delivery

Forecasting - ETC

- ETC based on new estimate
- ETC based on atypical variances
 - $ETC = BAC - EV$
- ETC based on typical variances
 - $ETC = (BAC - EV) / CPI$
- ETC based on both the CPI & SPI
 - $ETC = (BAC - EV) / (CPI * SPI)$

BAC = Budget at Completion
 BAC-EV = Remaining Work
 VAC = Variance at Completion

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

Forecasting - EAC

- Using a new estimate
– $EAC = AC + ETC$
- Using remaining budget
– $EAC = AC + (BAC - EV)$
- Using CPI
– $EAC = AC + ((BAC - EV) / CPI)$
- Using both CPI & SPI
– $EAC = AC + ((BAC - EV) / (CPI * SPI))$



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Value-Driven Delivery

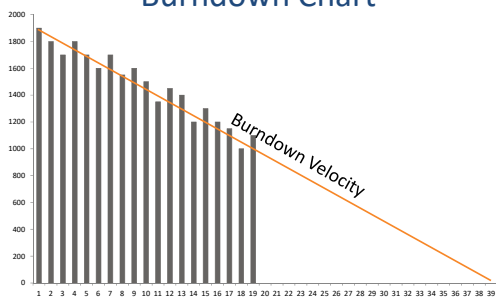
Forecasting - TCPI

- The calculated projection of cost performance that must be achieved on the remaining work to meet a specified management goal.
- Using BAC
– $TCPI = (BAC - EV) / (BAC - AC)$
- Using EAC
– $TCPI = (BAC - EV) / (EAC - AC)$
- $VAC = BAC - EAC$

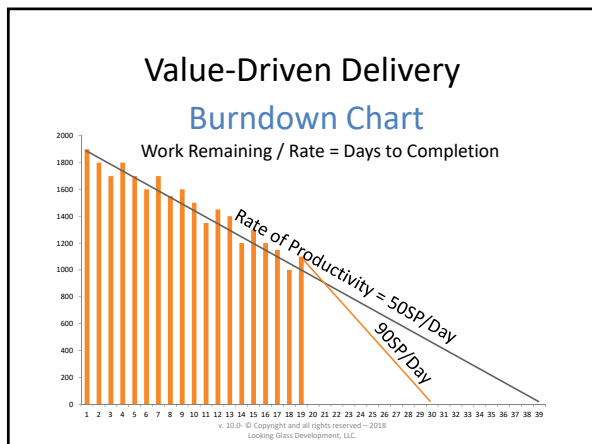
v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

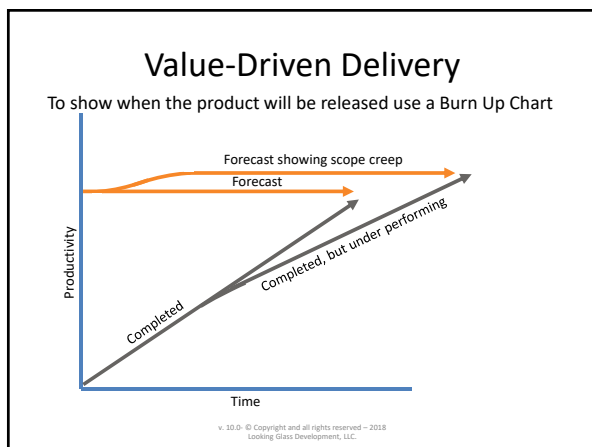
Value-Driven Delivery

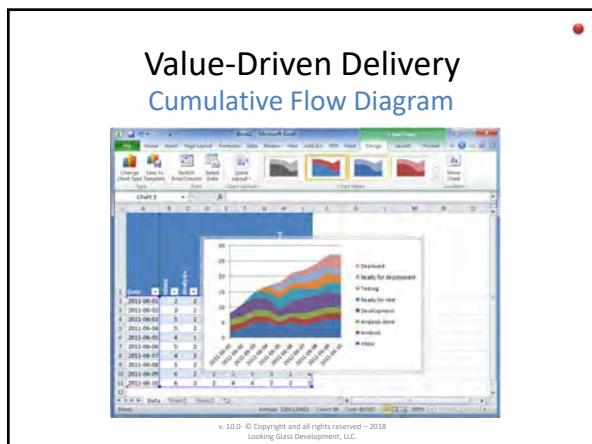
Burndown Chart



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.







Stakeholder Engagement

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

Domain Tasks – Stakeholder Engagement

1. Identify & engage effective & empowered business stakeholder(s) through periodic reviews in order to ensure that the team is knowledgeable about stakeholders' interests, needs, & expectations.
2. Identify & engage all stakeholders (current & future) by promoting knowledge sharing early & throughout the project to ensure the unimpeded flow of information & value throughout the lifespan of the project.

Domain Tasks – Ensure Stakeholder Involvement

3. Establish stakeholder relationships by forming a working agreement among key stakeholders in order to promote participation & effective collaboration.
4. Maintain proper stakeholder involvement by continually assessing changes in the project & organization in order to ensure that new stakeholders are appropriately engaged.
5. Establish collaborative behaviors among the members of the organization by fostering group decision making & conflict resolution in order to improve decision quality & reduce the time required to make decisions.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

Domain Tasks – Manage Stakeholder Expectations

6. Establish a shared vision of the various project increments (products, deliverables, releases, iterations) by developing a high level vision & supporting objectives in order to align stakeholders' expectations & build trust.
7. Establish & maintain a shared understanding of success criteria, deliverables, & acceptable trade-offs by facilitating awareness among stakeholders in order to align expectations & build trust.
8. Provide transparency regarding work status by communicating team progress, work quality, impediments, & risks in order to help the primary stakeholders make informed decisions.
9. Provide forecasts at a level of detail that balances the need for certainty & the benefits of adaptability in order to allow stakeholders to plan effectively.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement Who is a Stakeholder?

- Anyone with a vested interest in the project

Stakeholder Examples

Your Boss	Shareholders	The Government
Senior Executives	Alliance Partners	Trade Associations
Your Coworkers	Suppliers	The Press
Your Team	Lenders	Interest Groups
Customers	Analysts	The Public
Prospects	Future Employees	The Community
Your Family	Users	Sponsors

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

- Key Aspects
- Get the right stakeholders
- Maintain their involvement
- Actively manage their interest
- Frequently discuss “DoD”
- Show progress and capabilities
- Candidly discuss estimates and projections

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

Wireframes

- Used for quick mock-ups.
- Visual tool for stakeholders.
- Low fidelity tool – quick & cheap.



v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

Personas

- Provide an archetypal description of users.
- Are grounded in reality.
- Generate focus.
- Are tangible and actionable.
- Are goal oriented, specific and relevant.
- Do not replace requirements, simply augment them.
- Allow developers to empathize with users.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

Personas

- 1st introduced by Alan Cooper.
- Software needs to be designed for a specific person.
- Personas are different than Roles from Use Cases.
 - A customer is a role
 - A customer who is named, has specific circumstances and needs is a persona.
- Testing requires multiple personas.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

Persona Example: Frances Miller

- Sixty-seven year-old Frances is the mother of four children and the grandmother of twelve. She lives in her own home, bakes a pie once a week so that she has something to serve for Sunday visitors (usually one of her children and their immediate family), and has two cats. The cats' names are Fred and Wilma, names given to them by four-year old grandson Bobby. She likes to knit and do needlework, which she either gives away as presents to her family or donates to the annual sale to raise money for the church she belongs to.
- Every morning she goes for a one hour walk along the lake front when the weather is good. On bad days she'll go with her neighbor to the local mall where a group of senior citizens "Mall Stroll" each morning before sitting down at one of the restaurants for coffee or tea. For breakfast Frances prefers a cup of Earl Grey tea and two slices of whole-wheat toast with her own home-made preserves. Lunch is typically a bowl of soup or a sandwich and then she'll have the opposite for dinner.
- She is a middle-class retiree living on a fixed income and has been a widow for ten years. Her mortgage has been paid off and she has one credit card which she seldom uses. She has been a customer of the bank for 57 years although has never used an automated teller machine (ATM) and never intends to. She has no patience for phone banking and does not own a computer. Every Monday at 10:30 am she will visit her local bank branch to withdraw enough cash for the week. She prefers to talk with Selma the branch manager or with Robert, a CSR who was a high-school friend of her oldest son.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement User Stories

- User Stories represent features written from perspective of end users are called User Stories.
- A collection of User Stories is called the product backlog.
- User Story estimates are built using Story Points, but Story Points DON'T answer the question, when will my product ship?

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Stakeholder Engagement User Stories

- User Stories always use the form, "As a... (ROLE) I need to ... (FUNCTION) so that I may ... (SUCCESS CRITERIA).
- User Stories are always written in the language of the business.
- Each User Story MUST have acceptance criteria & be testable.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Stakeholder Engagement User Story Strengths

- Encourages effective communication about requirements through frequent face-to-face interactions.
- Avoids too much detail which can increase cost without any benefit [Forrester & Simula].
- Better adaptation to change.
- Acknowledges the fact that all requirements are NOT known at the beginning.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Stakeholder Engagement

User Stories The 3 Cs

- **Card** – The brief description must have meaning to both the team and the product owner.
- **Conversation** – This is the most important part. The card is not enough to write code. The card leads to a conversation to ensure understanding.
- **Confirmation** – This is the success criteria. It gives us the high-level criteria against which the resulting feature will be tested.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

You Must INVEST in Your Stories

- **I**ndependent – From other stories.
- **N**egotiable – Not set in stone.
- **V**aluable – Don't do it if it's not necessary.
- **E**stimatable – You must be able to come up with realistic numbers.
- **S**mall – Size matters!
- **T**estable – You must be able to prove it works.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

- Given, When, Then - Another format for stories.
- Used for non-functional or system based stories.
- Example:
 - **Given** the account is valid and the account has a MovieCredit balance of greater than \$0,
 - **When** the user redeems credit for a movie,
 - **Then** issue the movie and reduce the user's MovieCredit balance.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

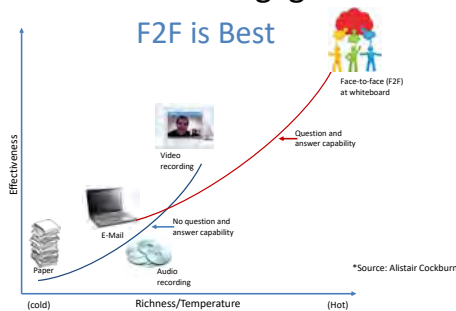
Definition of Done (DoD)

- Before anything is declared done...
 - **Tested** – Are all unit, integration and customer tests finished?
 - **Coded** – Has all the code been written?
 - **Designed** – Has the code been refactored to the team's satisfaction?

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

F2F is Best



*Source: Alistair Cockburn
v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

- **Information Radiators** – A number of highly visible methods to display information including large charts, graphs, & summaries of project data.
- Sometimes called “Visual Controls”
- From Alistair Cockburn

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

Information Radiators Examples

- Average Cycle Time Charts
- Burndown or Burn-Up Charts
- Cumulative Flow Diagram
- EVMS Diagram
- Velocity Tracking Chart – A bar or line graph displaying daily velocity.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

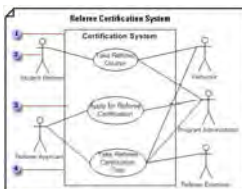
Agile Modeling

- Lightweight, barely sufficient capturing the design without a need for further polish.
- Scott Ambler top expert – Agile modeling's value peaks earlier than traditional theory leads us to believe.
- Types:
 - Use case diagrams
 - Data models
 - Screen designs

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

- The Diagram
- The Write Up



v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

- Soft Skills
- Negotiation
- **Active listening** – hearing what someone is really trying to convey rather than just the words.
Three levels of active listening:
 - **Level 1 Internal Listening** – Ask how is this going to affect me?
 - **Level 2 Focused Listening** – Put yourself in the mind of the speaker.
 - **Level 3 Global Listening** – Builds on level 2 to pick up on physical and environmental indicators.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

- Facilitation methods – Focus on:
 - Goals
 - Rules
 - Timing
 - Assisting
- Globalization, culture & team diversity
- Distributed teams
- Conflict resolution

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

Conflict Resolution

Resolution		Concern	
Mode	Style	Personal Goals	Relationships
• Withdrawal	• Lose - Leave	• Low	• Low
• Smoothing	• Yield – Lose	• Low	• High
• Compromising	• Compromise	• Medium	• Medium
• Forcing	• Win – Lose	• High	• Low
• Problem Solving (Confronting)	• Integrative	• High	• High

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Stakeholder Engagement

Speed B. Leas Conflict Model

Level	Name	Characteristics	Language Type	Atmosphere / Environment
Level 1	Problem to Solve	Information sharing & collaboration	Open & fact-based	<ul style="list-style-type: none"> People have different opinions or misunderstandings. Conflicting goals or values. Not comfortable, but not emotionally charged either.
Level 2	Disagreement	Personal protection trumps resolving the conflict	Guarded & open to interpretation	<ul style="list-style-type: none"> Self-protection becomes important. Team members distance themselves from the debate. Discussions happen off-line (outside the team environment) Good-natured joking moves to half-joking barbs.
Level 3	Contest	Winning trumps resolving the conflict	Includes personal attacks	<ul style="list-style-type: none"> The aim is to win. People take sides. Blaming flourishes.
Level 4	Crusade	Protecting one's own group becomes focus	Ideological	<ul style="list-style-type: none"> Resolving the situation is not good enough. Team members believe that people "on the other side" will not change and need to be removed.
Level 5	World War	Destroy the other!	Little or nonexistent	<ul style="list-style-type: none"> "Destroy!" is the battle cry The combatants must be separated No constructive outcome can be had

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Stakeholder Engagement

Participatory Decision Models

- Simple voting
- Thumbs up/down/sideways
- Highsmith's Decision Spectrum
- Fist-of-Five Voting – Likeart Scale

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Stakeholder Engagement

Management vs. Leadership

Management Focus	Leadership Focus
• Tasks/things	• People
• Control	• Empowerment
• Efficiency	• Effectiveness
• Doing things right	• Doing the right things
• Speed	• Direction
• Practices	• Principles
• Command	• Communication

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Stakeholder Engagement

Servant Leadership

- The practice of leading through service to the team, by focusing on understanding and addressing the needs and development of team members in order to enable the best team performance.
- A servant leader facilitates the team's discovery and definition of agile.
- Servant leaders practice and radiate agile.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Stakeholder Engagement

Servant Leader's Approach to Work

- **Purpose** – Work with the team to define the “why” or purpose so they can engage and coalesce around the goal for the project. The entire team optimizes at the project level, not the person level.
- **People** – Once the purpose is established, encourage the team to create an environment where everyone can succeed. Ask each team member to contribute across the project work.
- **Process** – Do not plan on following the “perfect” agile process, but instead look for the results. When a cross-functional team delivers finished value often and reflects on the product and process, the teams are agile. It does not matter what the team calls its process.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Stakeholder Engagement

Servant Leaders...

- Shields team from interruptions.
- Removes impediments to progress.
- (Re)communicates project vision.
- Carrys food & water.
- Manages relationships to build communication and coordination within the team and across the organization.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Stakeholder Engagement

Servant Leaders...

- Promoting self-awareness.
- Listening.
- Serving those on the team.
- Helping people grow.
- Coaching vs. controlling.
- Promoting safety, respect, and trust.
- Promoting the energy and intelligence of others.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Stakeholder Engagement

12 Principles for Leading Agile Projects

1. Learn the team members' needs.
2. Learn the project's requirements.
3. Act for the simultaneous welfare of the team and the project.
4. Create an environment of functional accountability.
5. Have a vision of the completed project.
6. Use the project vision to drive your own behavior.
7. Serve as the central figure in successful team development.
8. Recognize team conflict as a positive step.
9. Manage with an eye toward ethics.
10. Remember that ethics is not an afterthought, but an integral part of our thinking.
11. Take time to reflect on the project.
12. Develop the trick of think backwards.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Boosting Team Performance

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Team Performance

Domain Tasks – Team Formation

1. Cooperate with the other team members to devise ground rules & internal processes in order to foster team coherence & strengthen team members' commitment to shared outcomes.
2. Help create a team that has the interpersonal & technical skills needed to achieve all known project objectives in order to create business value with minimal delay.

Domain Tasks – Team Empowerment

3. Encourage team members to become generalizing specialists in order to reduce team size & bottlenecks, & to create a high-performing cross-functional team.
4. Contribute to self-organizing the work by empowering others & encouraging emerging leadership in order to produce effective solutions & manage complexity.
5. Continuously discover team & personal motivators & de-motivators in order to ensure that team morale is high & team members are motivated & productive throughout the project.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Team Performance

Domain Tasks – Team Collaboration & Commitment

6. Facilitate close communication within the team & the team & with appropriate external stakeholders through co-location or the use of collaboration tools in order to reduce miscommunication & rework.
7. Reduce distractions in order to establish a predictable outcome & optimize the value delivered.
8. Participate in aligning project & team goals by sharing the project vision in order to ensure the team understands how their objectives fit into the overall goals of the project.
9. Encourage the team to measure its velocity by tracking & measuring actual performance in previous iterations or releases in order for members to gain a better understanding of their capacity & create more accurate forecasts.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Boosting Team Performance

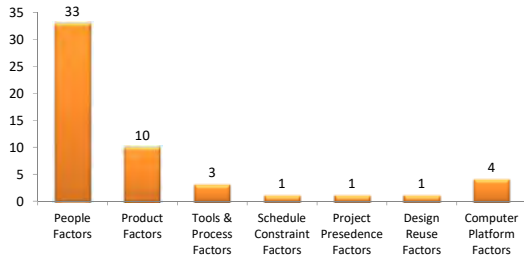
COCOMO

- Stands for Constructive Cost Model.
- Correlation study looking at thousands of software projects between input variables and total project costs.
- Results used as estimating technique.
- COCOMO II has the following weighting factors...

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Boosting Team Performance

Weighting Factors for COCOMO Input Variables



v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Boosting Team Performance

Adaptive Leadership

- Adapting how we lead based on circumstances.
- Five stages:
 - Forming
 - Storming
 - Norming
 - Performing
 - Adjourning or Mourning
- Bruce Tuckman model



v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Boosting Team Performance

Leadership Styles

- **Autocratic** – They solicit little or no informational input from their group and make managerial decisions solely by themselves.
- **Consultative Autocratic** – Intensive information input is solicited, but these leaders keep all substantive decision-making authority to themselves.
- **Consensus Manager** – They throw open the problem to the group and encourage the entire team to make the relevant decision.
- **Shareholder Manager** – (Poor Management) little or no information input and exchange takes place within the group context, yet the group is provided the authority for the final decision.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Boosting Team Performance

Theories of Management Style

- The Leadership Contingency Model (Fielder)
 - Holds that there is no best overall style.
 - Style is contingent on the situation.
 - Variables affecting the situation.
 - Team Leader ↔ Team Member relations
 - Degree of task structure
 - Position of power

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Boosting Team Performance

Theories of Management Style

- The Situational Leadership Theory (Hersey and Blanchard)
 - Identifies four (4) leadership styles
 - Delegating
 - Participating
 - Selling
 - Telling



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Boosting Team Performance

Emotional Intelligence

- The ability to identify, assess, and influence the emotions of ourselves, other individuals, and groups.



v. 10.0- © Copyright and all rights reserved – 2018 Looking Glass Development, LLC.

Boosting Team Performance

Ability-Based EI Model

- **Perceiving Emotions** – the ability to detect and decipher emotions in faces, pictures, voices, and cultural artifacts—including the ability to identify one's own emotions. Perceiving emotions represents a basic aspect of emotional intelligence, as it makes all other processing of emotional information possible.
- **Using Emotions** – the ability to harness emotions to facilitate various cognitive activities, such as thinking and problem solving. The emotionally intelligent person can capitalize fully upon his or her changing moods in order to best fit the task at hand.
- **Understanding Emotions** – the ability to comprehend emotional language and to appreciate complicated relationships among emotions. For example, understanding emotions encompasses the ability to be sensitive to slight variations between emotions, and the ability to recognize and describe how emotions evolve over time.
- **Managing Emotions** – the ability to regulate emotions in both ourselves and in others. Therefore, the emotionally intelligent person can harness emotions, even negative ones, and manage them to achieve intended goals.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Boosting Team Performance

Goleman's Mixed EI Model

- **Self-awareness** – the ability to know one's emotions, strengths, weaknesses, drives, values and goals and recognize their impact on others while using gut feelings to guide decisions.
- **Self-regulation** – involves controlling or redirecting one's disruptive emotions and impulses and adapting to changing circumstances.
- **Social skill** – managing relationships to move people in the desired direction.
- **Empathy** - considering other people's feelings especially when making decision.
- **Motivation** - being driven to achieve for the sake of achievement.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Boosting Team Performance

Empowered Teams

- Are self directing.
- Uses a servant leadership approach.
- "Team" is generally small – 10 to 20 people.
- Has complementary skills.
- Committed to a common purpose.
- Holds themselves mutually accountable.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Boosting Team Performance

High-Performance Teams

- Create a shared vision for the team.
- Set realistic goals.
- Limit team size to 12 or fewer members.
- Build a sense of team identity.
- Provide strong leadership.

Source: Frank LaFasto in Teamwork

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Boosting Team Performance

High-Performance Teams

- Are self-organizing.
- Are empowered to make decisions.
- Believe as a team they can solve any problem.
- Committed to team success.
- Owns its decisions and commitments.
- Trust motivates them.
- Consensus-driven with full divergence then convergence.
- Live in a world of constant constructive disagreement.

Source: Lyssa Adkins

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Boosting Team Performance

The Five Dysfunctions of a Team



© Copyright 2002 by Table Group Inc. All rights reserved. Used by permission of Table Group Inc.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Boosting Team Performance The Daily Scrum

- Stand up means STAND UP!
- Target 10 minutes, 15 max.
- Same time every day & don't miss a day.
- Stand in front of the visual progress artifact.
- Everybody is present.
- No typing during the meeting.
- Concentrate on the 2nd & 3rd questions.
- Don't talk to the ScrumMaster. Talk to the team.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Boosting Team Performance One-on-One Coaching & Mentoring

- Meet them half-step ahead.
- Guarantee safety.
- Partner with managers.
- Create positive regard.



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Boosting Team Performance Brainstorming Techniques

- Quiet Writing
- Round-Robin
- Free-for-All
- Adkins recommends "Green Zone / Red Zone" Model

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Boosting Team Performance

Green Zone

- Takes responsibility for circumstances of their life.
- Seeks to respond non-defensively.
- Seeks solutions rather than blame.
- Welcomes feedback
- Communicates a caring attitude.

Red Zone

- Blames others for the circumstances of their life.
- Feels threatened or wronged.
- Triggers defensiveness in others.
- Does not seek or value feedback.
- Communicates a high level of disapproval & contempt.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Boosting Team Performance

Other Tools

- Team Space
- **Co-located Teams** – Use the **Caves & Common Model** where offices are used for limited private conversations & most time spent in common areas.
- **Osmotic Communication** – Means that information flows in the background and is absorbed by the team members. They pick up relevant information by osmosis. Requires sitting in the same room.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

Domain Tasks – Levels of Planning

1. Plan at multiple levels (strategic, release, iteration, daily) creating appropriate detail by using rolling wave planning & progressive elaboration to balance predictability of outcomes with ability to exploit opportunities.
2. Make planning activities visible & transparent by encouraging participation of key stakeholders & publishing planning results in order to increase commitment level & reduce uncertainty.
3. As the project unfolds, set & manage stakeholder expectations by making increasingly specific levels of commitments in order to ensure common understanding of the expected deliverables.

Domain Tasks – Adaptation

4. Adapt the cadence & the planning process based on results of periodic retrospectives about characteristics &/or the size/complexity/criticality of the project deliverables in order to maximize the value.
5. Inspect & adapt the project plan to reflect changes in requirements, schedule, budget, & shifting priorities based on team learning, delivery experience, stakeholder feedback, & defects in order to maximize business value delivered.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

Domain Tasks – Agile Sizing & Estimation

6. Size items by using progressive elaboration techniques in order to determine likely project size independent of team velocity & external variables.
7. Adjust capacity by incorporating maintenance & operations demands & other factors in order to create or update the range estimates.
8. Create initial scope, schedule, & cost range estimates that reflect current high level understanding of the effort necessary to deliver the project in order to develop a starting point for managing the project.
9. Refine scope, schedule, & cost range estimates that reflect the latest understanding of the effort necessary to deliver the project in order to manage the project.
10. Continuously use data from changes in resource capacity, project size & velocity metrics in order to evaluate the estimate to complete.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

The Basic Agile Project Planning Process

Product Vision – Done Every Year.

Product Roadmap – Done every 6 months.

Release Planning – Done every 90 days.

Iteration Planning – Done every 2 to 6 weeks.

Daily Standup – Done every day.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

- Reducing non-value added work suggests planning only once.
- The only way to successfully plan a project is to plan and re-plan.
- This is adaptive planning
 - Accepting early plans are inaccurate.
 - Uncertainty drives the need to re-plan.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

Timeboxing

- Short, fixed duration periods of time in which activities or work are undertaken.
- Examples include:
 - The Daily Scrum Meeting is timeboxed to 15 minutes.
 - Iterations or sprints are timeboxed to 2 – 6 weeks.
- Agile locks the time leg of the triangle.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

Progressive Elaboration

- The process of capturing more detail over time as information emerges.
- Where is it used?
 - Estimates
 - WBS
 - Plans
 - Acceptance criteria
 - Test scenarios
- Two Types
 - Prototypes
 - Rolling Wave Planning

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

Rolling Wave Planning

- Detailed planning is done for activities in the near term & only high-level planning is done for the activities to be performed far into the future.
- The project is iteratively planned.
- It is a multi-step, intermittent process.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

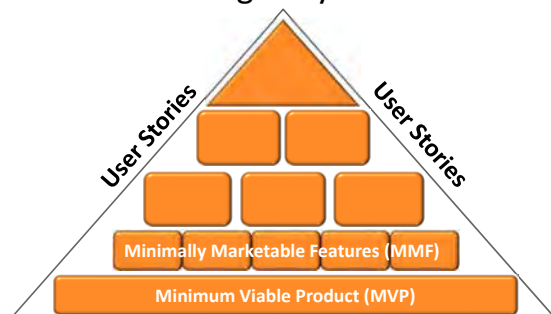
Adaptive Planning

Process Tailoring

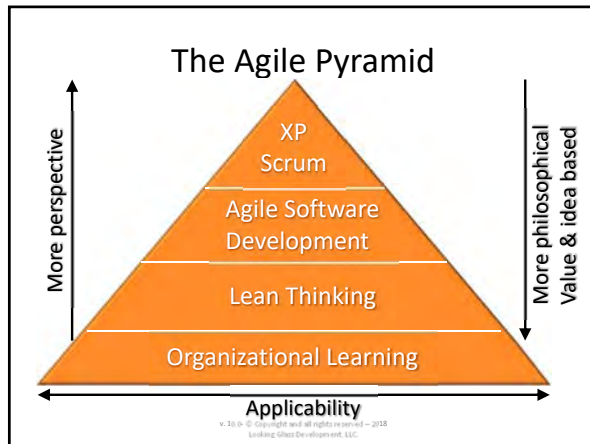
- Changing the process over time.
- Answering the questions:
 - What is going well?
 - What areas could use improvement?
 - What should we be doing differently?

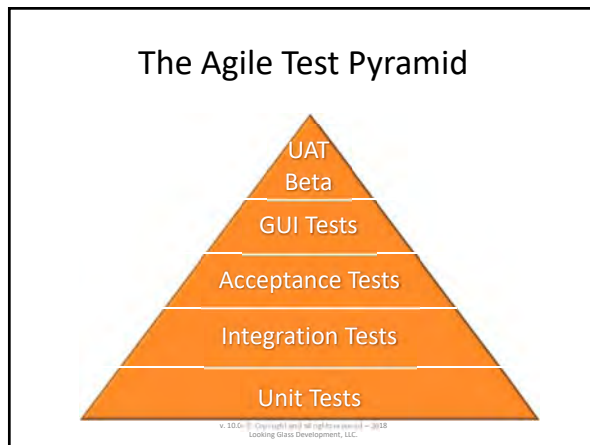
v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

The Agile Pyramid



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.





Adaptive Planning

Value-Based Analysis

- Process of considering the business value of work items & then acting accordingly.
- When prioritizing the work, the highest business value is done first.
- In Value-Based Analysis the costs must be offset.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Adaptive Planning

Value-Based Decomposition & Prioritization

- Eliciting requirements from stakeholders, ranking the requirements, then pulling prioritized requirements into the development process.
- Requirements are initially course grained.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

Agile Games

- Remember the Future
 - Each stakeholder starts independently.
 - Spend 20 minutes writing stickies on what world looks like.
 - Stickies brought together on the wall, duplicates are removed & items are grouped.
- Prune the Product Tree
 - Start by drawing a big tree.
 - Stakeholders add features as leaves to the tree.
 - Group leaves on branches.
 - Core features closer to the trunk.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

Agile Games

- Speedboat
 - A risk management exercise.
 - Done after Prune the Product Tree.
 - Draw picture of sailboat on water.
 - Positive risks are placed above water “wind”.
 - Negative risks placed below water “drag”.
 - Big negative risks can be “rocks”.
- Buy a Feature
- Bang for the Buck

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

Wideband Delphi & Planning Poker

- Wideband Delphi
 - Starts with group breaking down project into large components.
 - Group raises questions & discusses.
 - Anonymous estimates collected.
 - Aggregate results on plot.
 - Assumptions discussed & re-estimated.
 - Continue until reaching preset range.
- Planning Poker

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

Agile Estimation

- Agile estimation assumes complexity & uncertainty.
- Estimates are necessary for sizing & approvals, calculating ROI, etc.
- Holistic estimates require inclusion of development, rollout, & sustainment costs.
- Estimates should always be given as ranges.
- Estimation must be done continuously.
- Team members must do estimates.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

Estimation

- **Ideal Time** - Assumes resources 100% dedicated.
- **Relative Sizing** – Often called T-Shirt sizing.
- **Story Points** – Aggregates complexity & time.
- **Fibonacci Sequence** – Another comparison technique.
- **Affinity Estimating** – Process of grouping requirements into categories or collections. Used to group similarly sized user stories together.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

Time & Cost Estimation

- Determine the size
- Calculate the effort
- Convert effort into a schedule
- Calculate the cost



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

What Causes Project Delays?

- Unfocused management
- A focus on task management
- Dependencies between steps cause delays to accumulate and advances to be wasted.
- Parkinson's Law

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

What Causes Project Delays?

- The “*safety*” is misplaced
- Student Syndrome
- Lack of performance metrics
- Multi-tasking

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Adaptive Planning

Planning Differences

- Trial & demonstration uncover true requirements, which then require replanning.
- Agile planning is less of an upfront effort & is done more throughout the entire project.
- Midcourse adjustments are the norm.
- Business cases & charters are largely the same.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

Domain Tasks

1. Create an open & safe environment by encouraging conversation & experimentation, in order to surface problems & impediments that are slowing the team down or preventing its ability to deliver value.
2. Identify threats & issues by educating & engaging the team at various points in the project in order to resolve them at the appropriate time & improve processes that caused issues.
3. Ensure issues are resolved by appropriate team members &/or reset expectations in light of issues that cannot be resolved in order to maximize the value delivered.
4. Maintain a visible, monitored, & prioritized list of threats & issues in order to elevate accountability, encourage action, & track ownership & resolution status.
5. Communicate status of threats & issues by maintaining threat list & incorporating activities into backlog of work in order to provide transparency.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

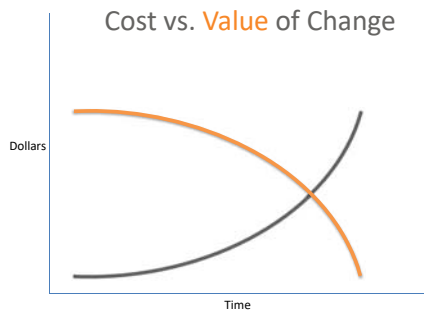
Problem Detection & Resolution

- **Cycle Time** – How long it takes to get things done.
- Cycle Time is closely related to WIP.
 - WIP is money spent with no return.
 - WIP hides bottlenecks & efficiency issues.
 - WIP represents potential rework.
- Agile attempts to reduce WIP & cycle times

$$\text{Cycle Time} = \frac{\text{WIP}}{\text{Throughput}}$$

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

Escaped Defects

- Daily standup used to identify most defects.
- Those missed that make it to the customer are called Escaped Defects.
- Escaped Defects are the most costly to fix.
- Escaped Defect Found Metric should scale down over time.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

Quality Standards

- Focus is “Fitness for Purpose” – What will the team do to ensure the quality and value of the product?
- Common actions:
 - Measuring tests passed & customer acceptance.
 - Automating tests.
 - Constant testing.
 - Ensure testers & developers collaborate.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

Failure Modes & Alternatives

- Making mistakes
- Preferring to fail conservatively
- Inventing rather than researching
- Being creatures of habit
- Being inconsistent

Alistair Cockburn

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

Failure Modes & Alternatives

- Be good at looking around
- Be able to learn
- Be malleable
- Take pride in your work

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

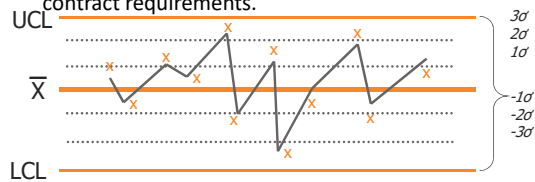
Failure Modes & Alternatives

- Counter with discipline & tolerance
- Start with something concrete and tangible
- Copy & alter
- Watch & listen
- Support concentration & communication
- Create personality matched assignments
- Use talent
- Create rewards that preserve joy
- Combine rewards
- Provide lots of feedback

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

- **In Control** – When ‘in control’ a process should not be adjusted.
- **Specification Limits** – Customer expectations or contract requirements.



- What is the ‘rule of 7’?

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

Continuous Integration

- Source code control system
- Build tools
- Test tools
- Scheduler or trigger
- Notifications

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

Continuous Integration

- Why use it?
 - Team receives early warning of bad or broken code.
 - Integration problems are fixed as they occur.
 - The team receives immediate feedback.
 - Ensures frequent unit testing.
 - Code can be reverted back quickly.
- Costs
 - Setup time.
 - Hardware to act as a build server.
 - Time to create automated testing.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

Risk-Based Spike

- A short sprint undertaken by the team to investigate an issue.
- Can result in a solution, recommendation or decision.
- Follows notion of failing fast.
- Used to test new technology early.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Test Driven Development (TDD)

- Tests are written **BEFORE** the code.
- A Unit Test is a test of a small, functional piece of code.
- Unit Tests are given priority in TDD.
- Unit tests make it...
 - Easier to find bugs.
 - Easier to maintain the code, but not test maintainability or test readability.
 - Easier to have full code coverage.
 - Easier to design & develop code.
 - Easier to deliver early & often.
 - Easier to track performance.

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.

Test Driven Development (TDD)

Traditional Coding Model

Step 1: Write Function

```
Double calculateLoan (int amount,  
int months, float interest rate)  
{  
  // ...assume functionality  
  // ...  
}
```

Step 2: Write test(s)

```
calculateLoan (20000,60,5.0);  
// is result 382.02  
  
CalculateLoan(6000,12,10.0);  
// is result 527.50?
```

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Test Driven Development (TDD)

The TDD Model

Step 1: Write test(s)

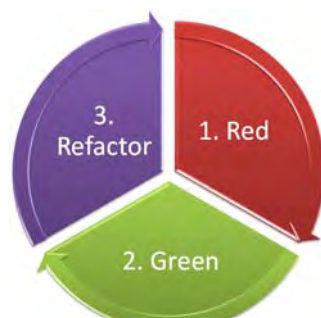
```
calculateLoan (20000,60,5.0);  
// is result 382.02  
  
CalculateLoan(6000,12,10.0);  
// is result 527.50?
```

Step 2: Write Code

```
Double calculateLoan (int amount,  
int months, float interest rate)  
{  
  // ...assume functionality  
  // ...  
}
```

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Test Driven Development (TDD)



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Test Driven Development (TDD)

- Must be able to make it fail. No code can be written without a failing test. This mean we actually run the tests to ensure the failure state before writing any code.
- Running the test to prove failure is a fundamental difference of TDD.
- Make it work. Code must be as simple as possible. The code must **ONLY** pass that new test for which it was designed.
- Make it better. This means you must refactor.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Test Driven Development (TDD)

- **TDD doesn't work for everything.**
 - Remember these are automated unit tests. Things like security, multi-threading, UI, game development issues might require more.
 - Unit testing does NOT replace other types of testing.
- **Do NOT write ALL the tests first.**
 - In the process write a single test, fail that test, write the code to pass that test. These are very small steps.
- **Separate tester or QA dept. does not write these tests.**
 - A separate QA dept. should be doing a different level of testing. It just slows down the process. It is not the same as TDD.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

Test-Driven Development

- **Advantages**
 - Focuses developer on needs of the customer.
 - Ensures at least some test are in place.
 - Helps catch defects early in cycle.
 - More modular, flexible & extendable.
- **Disadvantages:**
 - Unit tests usually done by developer.
 - Some functionality difficult to test with unit testing.
 - Tests must be maintained.
 - Higher number of passed tests equal false sense of success.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

Acceptance Test-Driven Development

- Also called ATDD.
- Moves testing focus from code to business requirements.
- Tests created before coding.
- Might use functional test framework such as FIT (Framework for Integrated Testing) or FitNesse.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

Acceptance Test-Driven Development

- Four stages:
 - **Discuss** the requirements – during planning meeting ask acceptance criteria.
 - **Distill** tests in a framework-friendly format.
 - **Develop** the code and hook up the tests.
 - **Demo** through exploratory testing.
- Regardless of method team must think about how the system will be tested before coding.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

TDD vs. Test 1st

- | | |
|-------------------------------|-------------------------------|
| • Automated | • Automated |
| • Functional Testing | • Functional Testing |
| • Testing by Programmers | • Testing by Programmers |
| • Testing @ Unit Level | • Testing @ Unit Level |
| • Testing Written Before Code | • Testing Written Before Code |
- In TDD code refactored to improve design. Skipped in Test 1st.
 - Test 1st says nothing about other activities in development cycle.
 - BIG DIFFERENCE with TDD – In TDD tests drive the design.

v. 10.0 - © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Extreme Programming

Refactoring

- **Refactoring** – Process of changing existing code to improve the way it functions.
- In XP you are not afraid of refactoring.
- Refactoring is part of your regular work & not a separate task.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Extreme Programming

Types of Refactoring

- **Yuck** – You look at code and it works, but is unsatisfactory. This is about making small improvements.
- **The Not Understood** – Code that you look at and cannot understand what it is doing. You must make code easier to understand.
- **New Insights** – When new functionality needs to be added, or you learn something.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Extreme Programming

Types of Refactoring

- **Planned Refactoring** – Actually adding refactoring to your project plan as a deliverable.
 - M. Fowler says it should hardly ever be done, because it represents a failure of the team to do the refactoring in small enough pieces to be constant.
 - Planned refactoring almost always requires justification.
 - Is evidence that you are not doing enough of the other types of refactoring.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Extreme Programming

Types of Refactoring

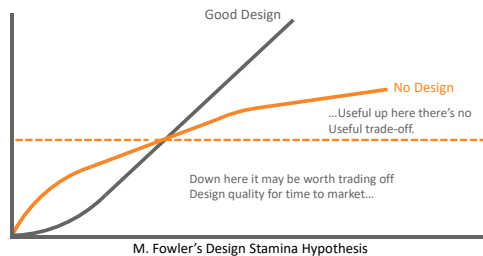
- **Long Term Refactoring** – Trying to get closer to some large future goal. Get some vision of where you want things to be in the future.
 - Must be done gradually.
 - Does not require significant planning.
 - The essence is doing small steps.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Extreme Programming

Refactoring

- Is refactoring just wasteful rework?

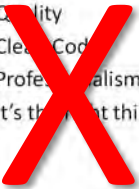


v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Extreme Programming

Why Refactor?

- Quality
- Clear Code
- Professionalism
- It's the right thing to do
- **Simple Economics** – It allows you to deliver more functionality more quickly and is the only reason you should be refactoring.



Problem Detection & Resolution

Problem Solving

- Basic problem solving
 - Gather data
 - Generate insights
 - Decide what to do

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

Problem Solving – Gather Data

- **Timeline** – Think about the project from a time-based perspective.
- **Triple Nickels** – Top 5 ideas are elaborated by 5 groups 5 times.
- **Color Code Dots** – Group affinity clustering technique.
- **Mad, Sad, Glad** – Explore the emotive elements.
- **Locate Strengths** – Look at what went well.
- **Satisfaction Histogram** – Review feelings through the project.
- **Team Radar** – Multidiscipline assessment tool.
- **Like to Like** – A strength diagnostic tool.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

Problem Solving – Generate Insights

- Brainstorming
- Five whys
- Fishbone
- Prioritize with dots
- Identify themes



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

5 Whys

- Used to determine the root cause of problems
- Developed by Sakichi Toyoda
- 2 Techniques used:
 - Fishbone Diagrams
 - Tabular Format



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Problem Detection & Resolution

Problem Solving - Decide What to Do

- Short Subjects
- SMART goals – Specific, Measurable, Attainable, Relevant, & Timely
- Retrospective Planning Game
- Circle of Questions

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Continuous Improvement

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Continuous Improvement

Domain Tasks


1. Tailor & adapt the project process by periodically reviewing & integrating team practices, organizational culture, & delivery goals in order to ensure team effectiveness within established organizational guidelines & norms.
2. Improve team processes by conducting frequent retrospectives & improvement experiments in order to continually enhance the effectiveness of the team, project, & organization.
3. Seek feedback on the product by incremental delivery & frequent demonstrations in order to improve the value of the product.
4. Create an environment of continued learning by providing opportunities for people to develop their skills in order to develop a more productive team of generalizing specialists.
5. Challenge existing process elements by performing a value stream analysis & removing waste in order to increase individual efficiency & team effectiveness.
6. Create systemic improvements by disseminating knowledge & practices across projects & organizational boundaries in order to avoid re-occurrence of identified problems & improve the effectiveness of the organization as a whole.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Continuous Improvement

Retrospectives Are Key

- Improved productivity
- Improved capability
- Improved quality
- Improved capacity



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Continuous Improvement

Retrospectives Steps

- Set the stage
 - Check-in
 - Focus on/focus off
 - ESVP – Participants anonymously associate themselves with an identity on slip of paper Explorer, Shopper, Vacationer, Prisoner
 - Working Agreements
- Gather data
- Generate insights
- Decide what to do
- Close the retrospective
 - Plus/delta – Franklin T
 - Helped, hindered, hypothesis
 - Return on time invested
 - Appreciations

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Continuous Improvement

Pre-Mortem

- Retrospective tool to solve problems BEFORE project is complete.
- Rules:
 - Set aside 2 hours of uninterrupted time.
 - All stakeholder MUST be present.
 - The Pre-Mortem must be a face-to-face meeting.
 - One person does nothing but take notes.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Continuous Improvement

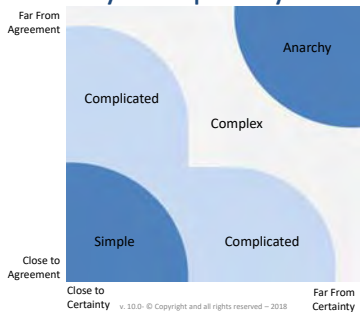
Pre-Mortem Process

- Spend 1st hour listing every possible problem.
- Pick top 10 problems.
 - Focus on show-stoppers.
 - Pick problems likely to happen.
 - Discard problems you have no control over.
- Spend the second hour creating solutions.
 - Create a proactive solution for problems.
 - Define a backup plan.

v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

Continuous Improvement

Stacey Complexity Model



v. 10.0- © Copyright and all rights reserved – 2018
Looking Glass Development, LLC.

ACP Exam Prep

v. 10.0 - © Copyright and all rights reserved - 2018
Looking Glass Development, LLC.
