Scrum Development Process

Lecturer: Ngo Huy Bien
Software Engineering Department
Faculty of Information Technology
VNUHCM - University of Science
Ho Chi Minh City, Vietnam
nhbien@fit.hcmus.edu.vn

Objectives

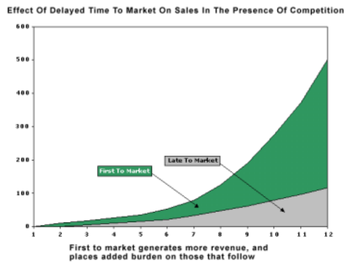
- To present *Agile development* concepts
- To present Scrum *roles*
- To present Scrum *activities*
- To present Scrum *products*
- To *apply* Scrum method to develop a software system

References

1. James Martin. Rapid application development. 1991.
2. Craig Larman, *Agile and Iterative Development: A Manager's Guide*. 2003.
3. Ken Schwaber. SCRUM Development Process. 1995.
4. Ken Schwaber, *Agile Project Management with Scrum*. 2004.
5. Jeff Sutherland and Ken Schwaber. The Scrum Papers -- Nuts, Bolts, and Origins of an Agile Process. 2007.
6. Sridhar Nerur et al. Challenges of Migrating to Agile Methodologies. 2005.
7. Brian Fitzgerald et al. Customising Agile Methods to Software Practices at Intel Shannon. 2006.
8. Jonathan Rasmusson. The Agile Samurai - How Agile Masters Deliver Great Software. 2010.

Time To Market

Time to market is the time until your product is sufficiently debugged that it can be shipped in volume production.



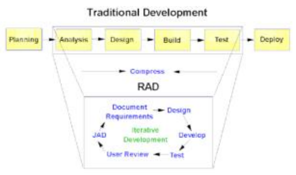
- Your Time To Market Determines The *Success* of Your Product
- Your Time To Market Determines Your *Rate of Return On Investment*

Rapid Application Development [1]

- *RAD* is an approach to building computer systems which combines
  - Computer-Assisted Software Engineering (CASE) tools and techniques,
  - user-driven prototyping, and
  - stringent project delivery time limits into a potent, tested, reliable formula for top-notch quality and productivity.
- RAD takes advantage of *automated tools* and techniques to restructure the process of building information systems.
- RAD replaces *hand-design* and *coding processes*, which are dependent upon the skills of isolated individuals, with automated design and coding, which is an inherently more stable process.

The RAD Approach

- RAD *compresses* the step-by-step development of conventional methods into an iterative process.
- The RAD approach thus includes *developing and refining* the data models, process models, and prototype in parallel using an iterative process.
- User requirements are refined, a solution is designed, the solution is prototyped, the prototype is reviewed, user input is provided, and the process begins *again*.



Essential Aspects of RAD

- Rapid Application Development has four essential aspects:
  - methodology,
  - people,
  - management, and
  - tools.
- If any one of these ingredients is *inadequate*, development will not be high speed.

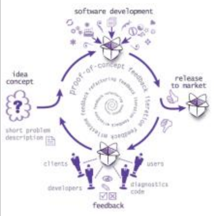
RAD
Methodology
People
Management
Tools

Lightweight Documentation

- Traditional system documentation
  - instantly *out of date*,
  - often *misleading* and
  - *expensive* to maintain
- Solution: making system knowledge *explicit*

Agile Software Development [2]

Agile development methods apply time-boxed iterative and evolutionary development, adaptive planning, promote evolutionary delivery, and include other values and practices that *encourage agility*—rapid and flexible response to change.

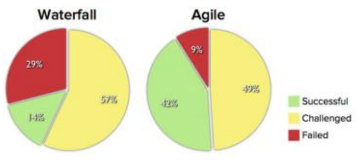


1. *Individuals and interactions* over processes and tools
2. *Working software* over comprehensive documentation (Produce no document unless its need is immediate and significant)
3. *Customer collaboration* over contract negotiation (A successful contract should govern the way the developing team and the customer collaborate rather than details of scope and schedule for a fixed cost.)
4. *Responding to change* over following a plan

Agile Principles

1) Early and continuous delivery of valuable software
2) Welcome changing requirements, even late in development
3) Deliver working software frequently with a preference to the shorter timescale
4) Business people and developers must work together daily
5) Trust individuals to get the job done
6) Face-to-face conversation
7) Working software is the primary measure of progress.
8) Sustainable development
9) Continuous attention to technical excellence and good design enhances agility
10) Simplicity is essential
11) The best architectures, requirements, and designs emerge from self-organizing teams
12) At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

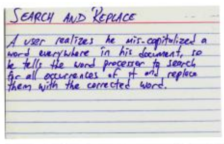
Evidence



Source: The CHAOS Manifesto, The Standish Group, 2012.

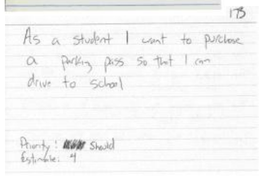
User Story

User stories are one-liners that state customer requirements.



SEARCH AND REPLACE

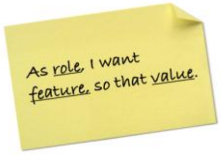
A user realizes he mis-capitalized a word everywhere in his document, so he tells the word processor to search for all occurrences of it and replace them with the corrected word.



173

As a student I want to purchase a parking pass so that I can drive to school

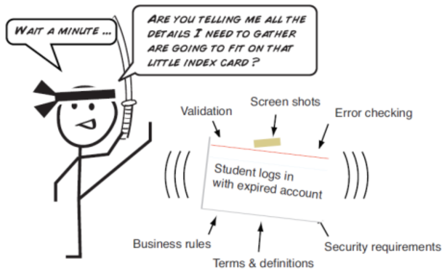
Priority: ~~[illegible]~~ Should
Estimate: 4



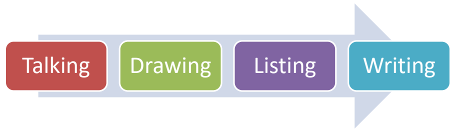
As a *role*, I want *feature*, so that *value*.

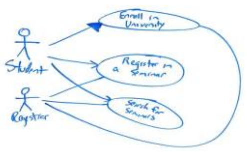
I – Independent
N – Negotiable
V – Valuable
E – Estimable
S – Small (+ Screens)
T – Testable

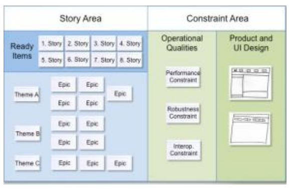
Index Cards Remind Us *NOT* to Try to Write Everything Down


WAIT A MINUTE ...
ARE YOU TELLING ME ALL THE DETAILS I NEED TO GATHER ARE GOING TO FIT ON THAT LITTLE INDEX CARD ?
Validation
Screen shots
Error checking
Student logs in with expired account
Business rules
Terms & definitions
Security requirements

Requirements Gathering

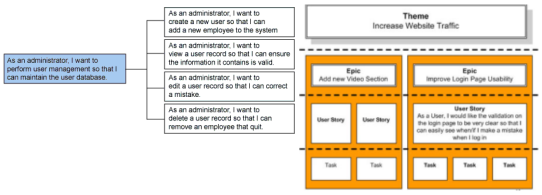






Epics, Themes and Tasks

- *Epics* are large user stories that need to be disaggregated into smaller user stories at some point.
- A *theme* is a group of epics.



As an administrator, I want to perform user management so that I can maintain the user database.

As an administrator, I want to create a new user so that I can add a new employee to the system

As an administrator, I want to view a user record so that I can ensure the information it contains is valid.

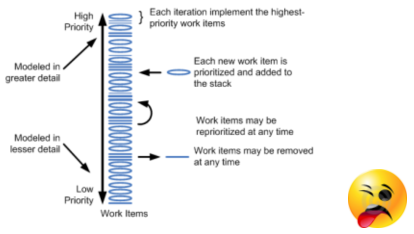
As an administrator, I want to edit a user record so that I can correct a mistake.

As an administrator, I want to delete a user record so that I can remove an employee that quit.

User Story
As a User, I would like the validation on the login page to be very clear so that I can easily see when/if I make a mistake when I log in

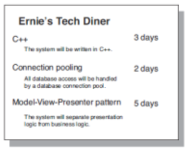
Product Backlog

The product backlog is the master list of *all functionality* desired in the product, prioritized as an absolute ordering by business value, containing *rough estimates* of both *business value*, frequency of use and *development effort*.

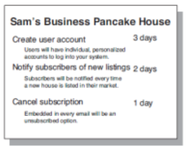

High Priority
Low Priority
Modeled in greater detail
Modeled in lesser detail
Work Items
Each iteration implement the highest-priority work items
Each new work item is prioritized and added to the stack
Work items may be reprioritized at any time
Work items may be removed at any time

We don't need a product backlog we just need to figure out what to do for the next Sprint!!!

Which Restaurant Would Your Hungry Customer Rather Dine At? [8]



Ernie's Tech Diner	
C++ The system will be written in C++.	3 days
Connection pooling All database access will be handled by a database connection pool.	2 days
Model-View-Presenter pattern The system will separate presentation logic from business logic.	5 days



Sam's Business Pancake House	
Create user account Users will have individual, personalized accounts to log into your system.	3 days
Notify subscribers of new listings Subscribers will be notified every time a new house is listed in their market.	2 days
Cancel subscription Embedded in every email will be an unsubscribed option.	1 day

Scrum 101

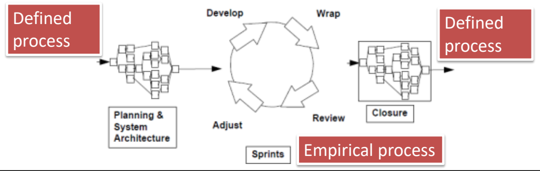
Kick-Off Meeting

Kick-off meeting is held to agree on the foundation objectives and requirements of the project.

- Project vision
- Feasibility study
- Statement of work
- Product backlog

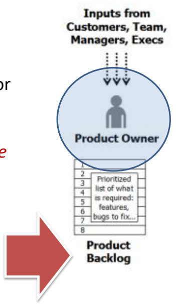
What is Scrum? [3]

- SCRUM is a management, enhancement and maintenance methodology for *an existing system* or *production prototype*.
- Waterfall and Spiral methodologies set the *context* and *deliverable definition* at the start of a project.
- SCRUM and Iterative methodologies initially plan the *context* and **broad** *deliverable definition*, and then **evolve** the deliverable during the project based on the environment.



The Product Owner

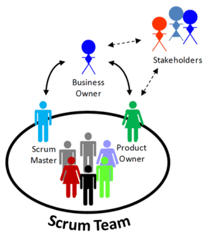
- Representing the *interests of everyone* with a stake in the project and its *resulting system*.
- Achieving initial and ongoing funding for the project by creating the project's initial *overall requirements*, *return on investment (ROI) objectives*, and *release plans*.
- Using the *product backlog* to ensure that the most valuable functionality is produced first and built upon.


Inputs from Customers, Team, Managers, Execs
Product Owner
1
2
3
4
5
6
7
8
Prioritized list of what is required: features, bugs to fix...
Product Backlog

The Team

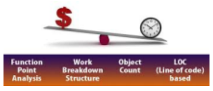
- *Developing* functionality
- *Self-managing*, self-organizing, and cross-functional

- The Scrum Master
  - Scrum *process*
  - *Teaching* Scrum to everyone involved in the project
  - Implementing Scrum so that it fits within an *organization's culture*
  - Ensuring that everyone follows Scrum *rules* and practices



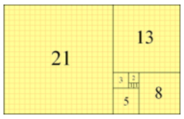
High Level Planning

- Development of a <u>comprehensive</u> *backlog list*.
- Create a high level estimates. *How*?



High Level Estimates

- 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, …

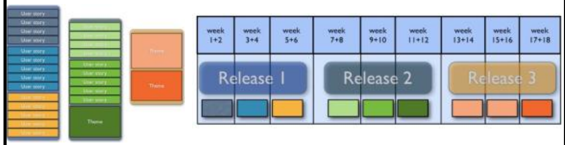


We CAN'T come up with accurate estimates!!!

- The key here is about *relativity*.
- Smallest thing: 1. Biggest thing: 21.
- Pick one *familiar* thing, give it 3, for example, based upon your experience. Pick another thing. Is it *bigger* or *smaller* than the previous one?

Release Planning

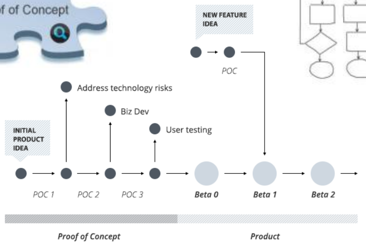
Release planning is a meeting used to create a release plan, which lays out the overall project.



How to Create A Release Plan?

- Günther Ruhe and Moshood Omolade Saliu. The Art and Science of Software Release Planning. 2005.

Scrum Releases (Roadmap)



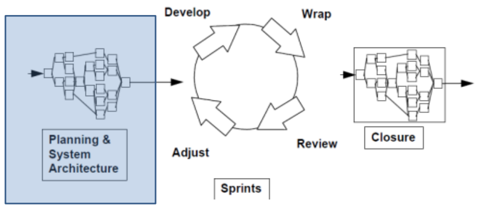
Release 101

Release Backlog

- Selection of the *release* most appropriate for immediate development.
- *Release backlog* are the user stories that are included in the next release.
- Mapping of *product packets* (objects) for backlog items in the selected release.

Product Backlog
Release Backlog

Pregame

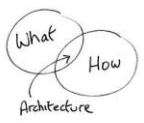


Planning

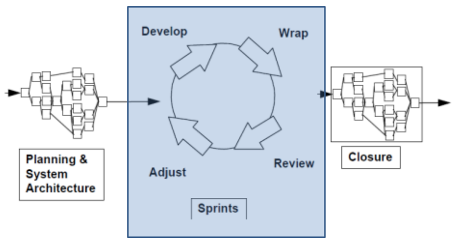
- Definition of *project team(s)* for the building of the new release.
- Assessment of risk and appropriate *risk controls*.
- Validation or reselection of *development tools* and infrastructure.
- Estimation of release *cost*, including development, collateral material, marketing, training, and rollout.
- Verification of *management approval* and funding.

System Architecture

- Review *assigned* backlog items.
- Identify changes necessary to *implement* backlog items.
- Perform *domain analysis* to the extent required to build, enhance, or update the domain models to reflect the new system context and requirements.
- Refine the *system architecture* to support the new context and requirements.
- Identify any *problems* or issues in developing or implementing the changes.
- Design *review meeting*, each team presenting approach and changes to implement each backlog item. Reassign changes as required.

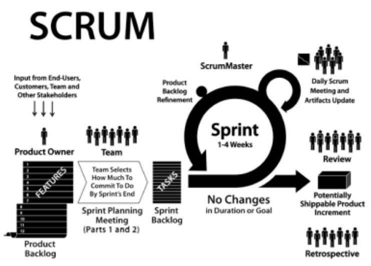


Game



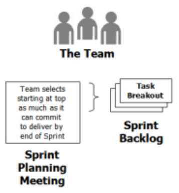
A Fist Sprint

With Scrum, projects progress via a <u>series of iterations</u> called *sprints*. Each sprint is typically *2-4 weeks* long.


SCRUM
Input from End-Users, Customers, Team and Other Stakeholders
Product Owner
Team
FEATURES
Team Selects How Much To Commit To Do By Sprint's End
Sprint Planning Meeting (Parts 1 and 2)
TASKS
Sprint Backlog
Product Backlog
ScrumMaster
Product Backlog Refinement
Sprint
1-4 Weeks
No Changes
in Duration or Goal
Daily Scrum Meeting and Artifacts Update
Review
Potentially Shippable Product Increment
Retrospective

Sprint Backlog

The *sprint backlog* is the list of work the team must address during the next sprint.


The Team
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog

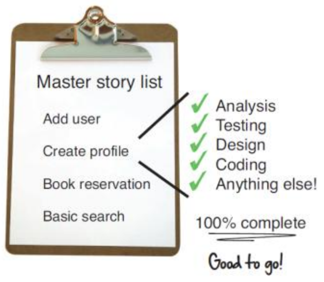
Sprint Planning Objectives

- Sprint deliverables
- *How* to achieve the sprint deliverables?

What Does "Done" Mean?

- Agreement between Product Owner and the Team.

Definition Of Done [8]

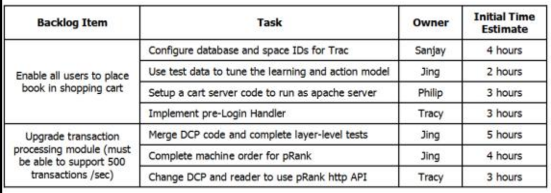


Example

- coded to standards
- reviewed by other member
- implemented with unit tests
- tested with 100 percent test automation
- integrated and deployed
- documented
- tested by other member
- accepted by PO

Sprint Planning [4] [5]

Features are broken down into tasks, which, as a best practice, should normally be between <u>four</u> and <u>sixteen</u> <u>hours</u> of work.



Backlog Item	Task	Owner	Initial Time Estimate
Enable all users to place book in shopping cart	Configure database and space IDs for Trac	Sanjay	4 hours
	Use test data to tune the learning and action model	Jing	2 hours
	Setup a cart server code to run as apache server	Philip	3 hours
	Implement pre-Login Handler	Tracy	3 hours
Upgrade transaction processing module (must be able to support 500 transactions /sec)	Merge DCP code and complete layer-level tests	Jing	5 hours
	Complete machine order for pRank	Jing	4 hours
	Change DCP and reader to use pRank http API	Tracy	3 hours

Individual Work [7]

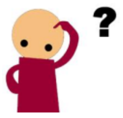
- Where available, *explicit process knowledge* is used; otherwise *tacit knowledge* and trial and error is used to build process knowledge.

Daily Meeting

1. What did you do since last Scrum meeting?
2. Do you have any obstacles?
3. What will you do before next meeting?

The team's ability to tackle its problems and solve them is the heart of Scrum

Sprint Practices



Sprint Review

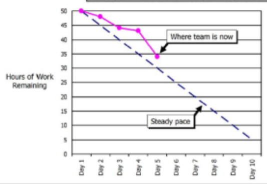
Sprint review is a meeting at after the Sprint ends, it's just a demo of what's been built, and anyone present is free to ask questions and give input.

How Can We Track Project Status?

- Scope (release) delivered?
- High-level plan (start date, end date/ estimated completion date, total effort, total duration, release dates)?
- Current release status (release date, % completed, remaining tasks)?
- Total budget (time) spent?
- Total remaining budget (time)?
- On track? Late? Fast?
- Risks?
- Issues?
- Scope changes? New estimate?

Burn-down Chart

Task	Task Owner	Hours of Work Remaining on Each Day of the Sprint									
		Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
Configure database and space IDs for Trac	Sanjay	4	4	3	1	0					
Use test data to tune the learning and action model	Jing	2	2	2	2	1					
Setup a cart server code to run as apache server	Philip	3	3	5	2	0					
Implement pre-Login Handler	Tracy	3	3	3	3	3					
Merge DCP code and complete layer-level tests	Jing	5	5	2	2	2					
Complete machine order for pRank	Jing	4	4	3	3	3					
Change DCP and reader to use pRank http API	Tracy	3	3	0	0	0					
	Total	50	48	44	43	34					

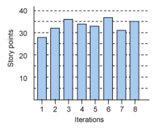


Burn-down chart shows, each day, *how much work* (measured in hours or days) *remains* until the team's commitment is completed.

Measuring Velocity

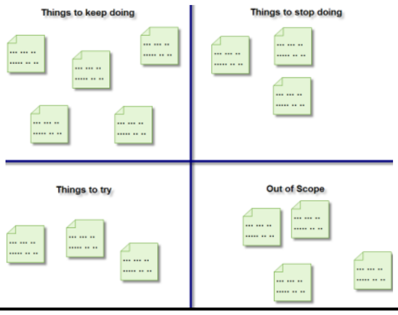
Iteration	Points
1	28
2	32
3	36
4	34
5	33
6	37
7	31
8	35

Velocity is the long-term tracking of how much work has been done by a team per iteration.

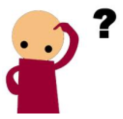


Sprint Retrospective

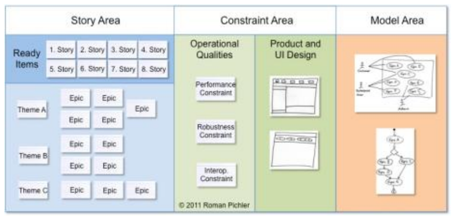
Sprint retrospective is a meeting for the team to discuss what's working and what's not working, and agree on changes to try.



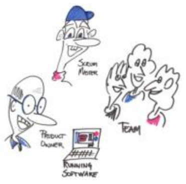
Where Do We Go Now?

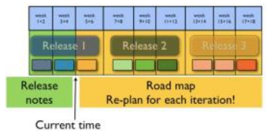


Here We Go: Release Backlog Board



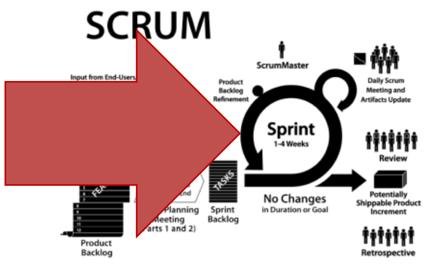
Here We Go: Re-planning





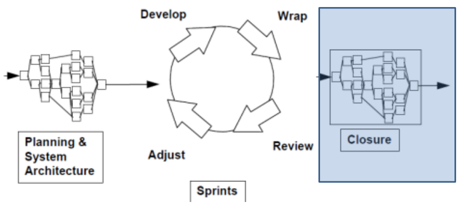
1. The schedule is *not changed*. The 4th iteration still ends with week number 8, and the releases are delivered in the weeks 6, 12 and 18.
2. The *scope* and content of the project *is adjusted empirically* as the developers and customers gain understanding of the solution.

And… A Second Sprint



SCRUM
Input from End-Users
ScrumMaster
Product Backlog Refinement
Daily Scrum Meeting and Artifacts Update
Sprint
1-4 Weeks
Review
No Changes
in Duration or Goal
Potentially Shippable Product Increment
Retrospective
Planning Meeting (Parts 1 and 2)
TASKS
Sprint Backlog
Product Backlog

Postgame

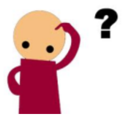


- This phase prepares the developed product for *general release*.
- Integration, system test, user documentation, training material preparation, and marketing material preparation are among *closure tasks*.

Tools

- https://www.atlassian.com/software/jira
- http://www.agilefant.com/
- https://trello.com/
- https://slack.com/

Scrum Work Products



How to Control the Project? [3]

- *Controls* in the SCRUM methodology are:
  - *Backlog*: Product functionality requirements that are not adequately addressed by the current product release.
  - *Release/Enhancement*: backlog items that at a point in time represent a viable release based on the variables of requirements, time, quality, and competition.
  - *Packets*: Product components or objects that must be changed to implement a backlog item into a new release.

Controls in the SCRUM Methodology Are Also

- *Risks*: risks that effect the success of the project are continuously assessed and responses planned.
- *Changes*: Changes that must occur to a packet to implement a backlog item.
- *Problems*: Technical problems that occur and must be solved to implement a change.
- *Solutions*: solutions to the problems and risks, often resulting in changes.
- *Issues*: Overall project and project issues that are not defined in terms of packets, changes and problems.

Controls Management

- These controls are used in the *various phases* of SCRUM.
- *Management* uses these controls to manage backlog.
- *Teams* use these controls to manage changes, problems.
- *Both* management and teams jointly manage issues, risks, and solutions.
- These controls are reviewed, *modified*, and reconciled at every Sprint review meeting.

In Short

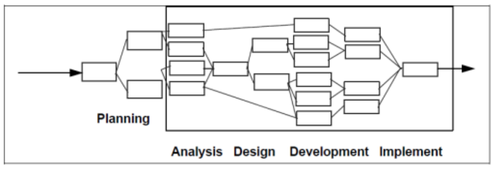
- The SCRUM methodology embodies these general, loose controls, using OO techniques for the actual *construction of deliverables*.

Scrum Values

- Commitment
- Focus
- Openness
- Respect
- Courage

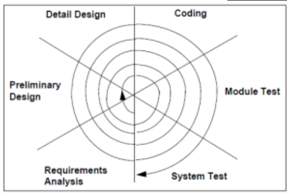
Waterfall Methodology Problem

- Its *linear* nature has been its largest problem.
- The process does not define how to respond to *unexpected output* from any of the intermediate process.

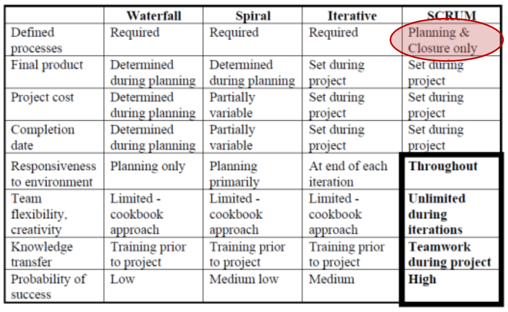


Iterative Methodology Problem

- The overall project deliverable has been *partitioned* into prioritized subsystems, each with clean interfaces.
- The Iterative approach still expects that the *underlying* development processes are defined and **linear**.

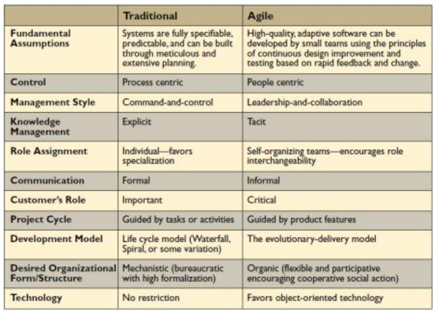


Methodology Comparison [3]



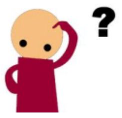
	Waterfall	Spiral	Iterative	SCRUM
Defined processes	Required	Required	Required	Planning & Closure only
Final product	Determined during planning	Determined during planning	Set during project	Set during project
Project cost	Determined during planning	Partially variable	Set during project	Set during project
Completion date	Determined during planning	Partially variable	Set during project	Set during project
Responsiveness to environment	Planning only	Planning primarily	At end of each iteration	**Throughout**
Team flexibility, creativity	Limited - cookbook approach	Limited - cookbook approach	Limited - cookbook approach	**Unlimited during iterations**
Knowledge transfer	Training prior to project	Training prior to project	Training prior to project	**Teamwork during project**
Probability of success	Low	Medium low	Medium	**High**

Traditional vs. Agile [6]



	Traditional	Agile
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning.	High-quality, adaptive software can be developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change.
Control	Process centric	People centric
Management Style	Command-and-control	Leadership-and-collaboration
Knowledge Management	Explicit	Tacit
Role Assignment	Individual—favors specialization	Self-organizing teams—encourages role interchangeability
Communication	Formal	Informal
Customer's Role	Important	Critical
Project Cycle	Guided by tasks or activities	Guided by product features
Development Model	Life cycle model (Waterfall, Spiral, or some variation)	The evolutionary-delivery model
Desired Organizational Form/Structure	Mechanistic (bureaucratic with high formalization)	Organic (flexible and participative encouraging cooperative social action)
Technology	No restriction	Favors object-oriented technology

Fixed-Price, Fixed-Date Contracts



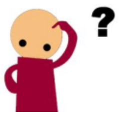
Some Techniques

- http://agilekiwi.com/estimationandpricing/creating-an-agile-contract
- https://en.wikipedia.org/wiki/Reference_class_forecasting

No Sustainable Pace

- Every sprint becomes a small two or four-week project with a well-defined scope, a clear beginning, and a *fixed end date*.
- Many teams were *not able to deliver* what they had agreed to at the start of their sprint.
- Reasons:
  - Wrong estimation
  - The unexpected
- Solutions:
  - Follow the life cycle: analysis, design, design test, build, developer test, other team member test, and acceptance test
  - Risk reserve

Capability Maturity Model Integration



Thank You for Your Time