AGILE SOFTWARE DEVELOPMENT

Keith Pine • Kumeel Alsmail • Parker Li • Björn Davis

INTRODUCTION TO AGILE

- What is Agile?
- Origins of Agile
- Does Agile Work?
- Methodologies

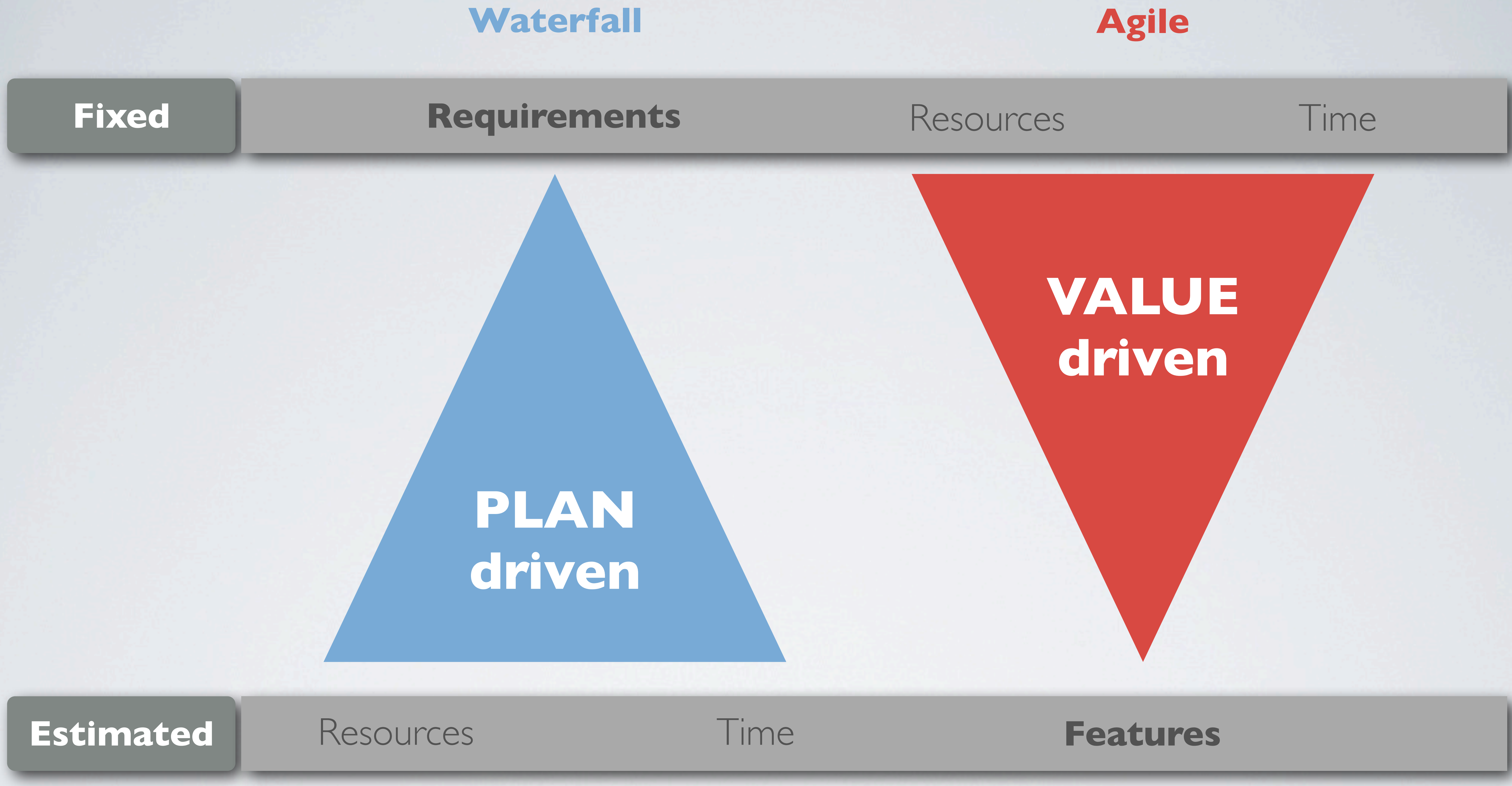
WHAT IS AGILE?

- A set of software methodologies that are based on iterative development
- Requirements are expected to change, and response to change is rapid
- Time-boxed releases

PLAN-DRIVEN VS ITERATIVE

- Plan-driven methods need requirements in advance
  - Waterfall
  - Requirements are relatively static
- Iterative methods expect change
  - Spiral-model
  - Evolutionary processes
  - and Agile

PROJECT CONSTRAINTS



Waterfall

Agile

Fixed

Requirements

Resources

Time

PLAN driven

VALUE driven

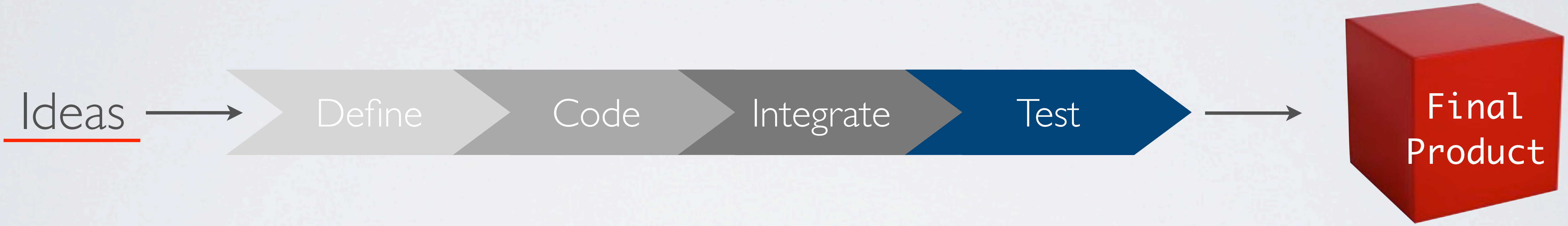
Estimated

Resources

Time

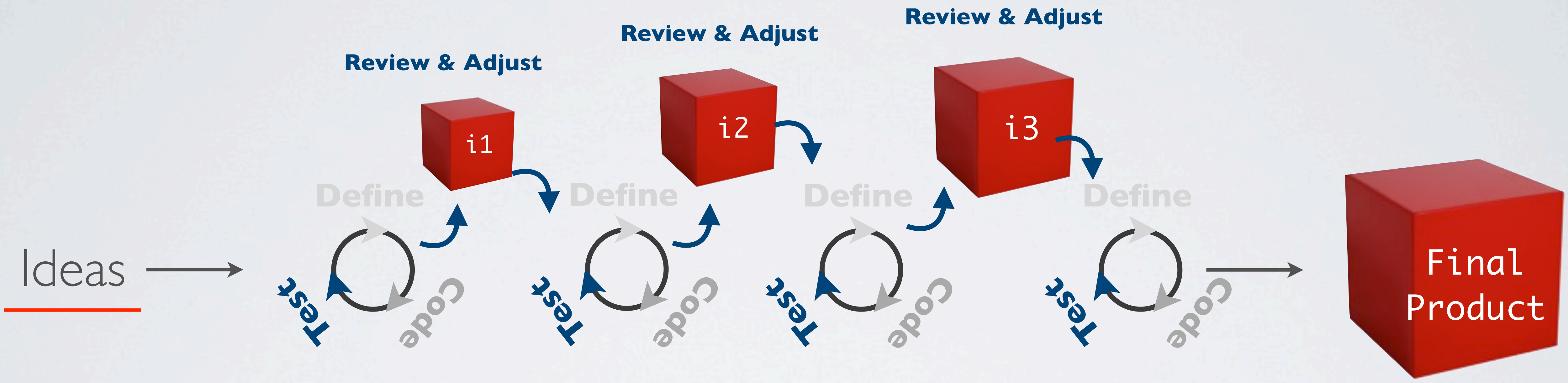
Features

TRADITIONAL DEVELOPMENT



Ideas → Define → Code → Integrate → Test → Final Product

AGILE DEVELOPMENT



PREDECESSORS OF AGILE

FDD

XP

Scrum

Crystal

ASD

DSDM

ORIGINS

- February 2001 at Snowbird Ski Resort
- 17 representatives of emerging ‘lightweight’ methods met to find a common ground
  - Kent Beck, Alistair Cockburn, Ward Cunningham, Martin Fowler, Jim Highsmith, Bob Martin, etc.
- Looking for freedom from “Dilbertesque corporations”
- The term “Agile” was coined

MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

12 PRINCIPLES

1. Customer satisfaction through early and continuous delivery of valuable software
2. Changing requirements are welcome, even late in development.
3. Deliver working software as quickly as possible
4. Daily cooperation between business people and developers
5. Build projects around motivated individuals, and provide the support they need
6. Face-to-face conversation is the most efficient and effective method of communication
7. Progress is primarily measured through working software
8. Sustainable development is promoted
9. Continuous attention to technical excellence and good design
10. Simplicity – the art of maximizing the amount of work not done – is essential
11. Self organizing teams produce the best architectures, requirements and designs
12. Regular reflection by the team is key to becoming more effective

PRINCIPLES NOT RULES

- Values and Principles
- Not a pre-defined process
- Context sensitive to each team and project
- Common recommendations
  - iterations no longer than a month long
  - each iteration should result in something that everyone agrees is "done" with feedback from the stakeholders
  - figure out what you're doing at the start of the iteration
  - reflect on accomplishments/failures at the end of the iteration
  - non-stop communication

BENEFITS OF AGILE

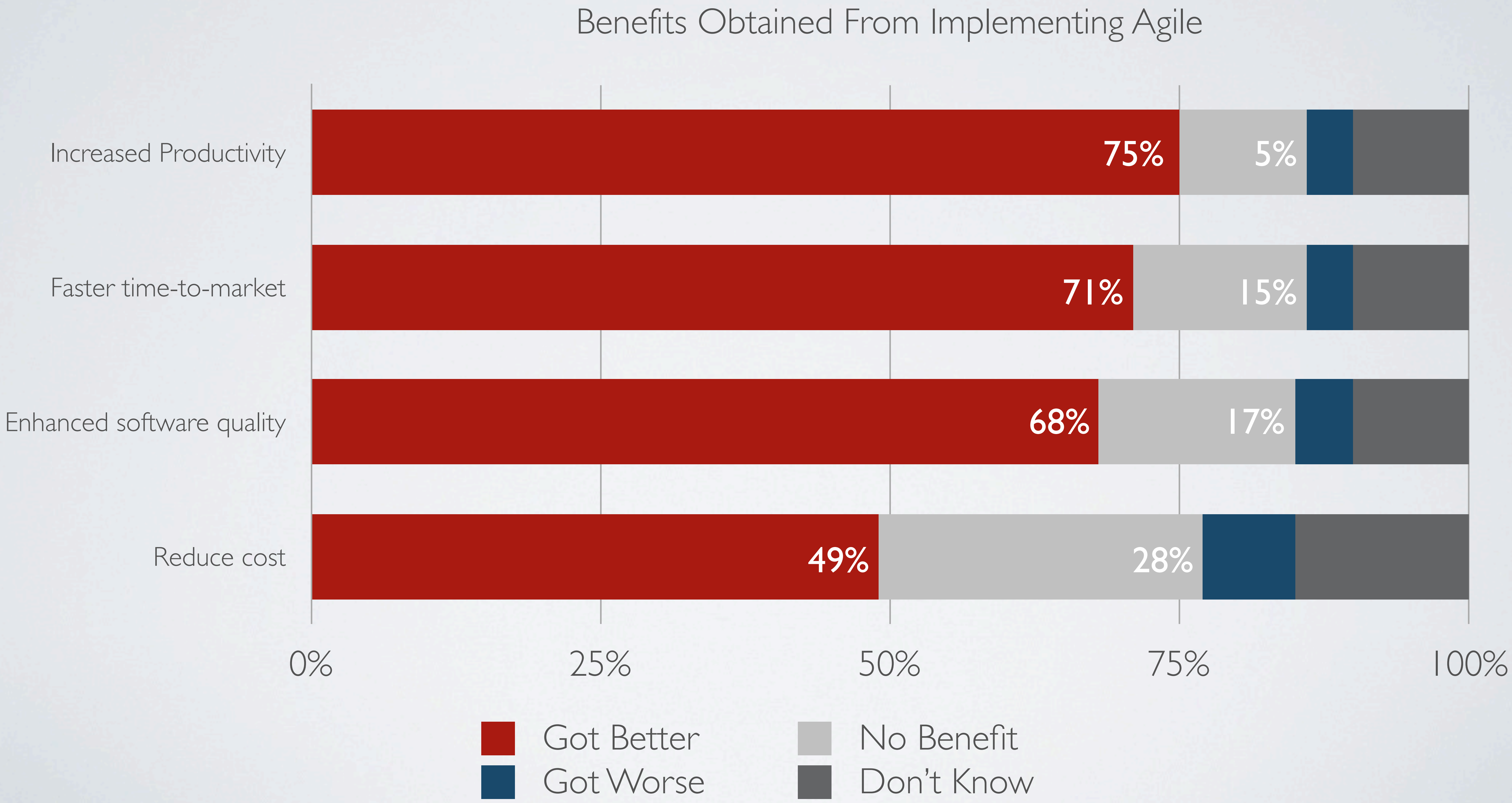
- Able to quickly respond to changing requirements and priorities
- Lower cost and risk
- Continuous integration and delivery - more visibility into project progress
- Business value is prioritized
- Delivers business value early and often

DOES AGILE WORK?

- Scott Ambler (Practice Leader for Agile Development with IBM) - Agile Adoption Rates Survey
- How have agile approaches affected your productivity?
  - 60% report increased productivity
  - 34% report no change
  - Only 6% claimed lowered productivity
- How have agile approaches affected the quality of systems deployed?
  - 66% responded that quality is higher
  - About 30% said no change in quality
- How have agile approaches affected business stakeholder satisfaction?
  - 58% reported improved satisfaction
  - 3% report reduced satisfaction.

DOES AGILE WORK?

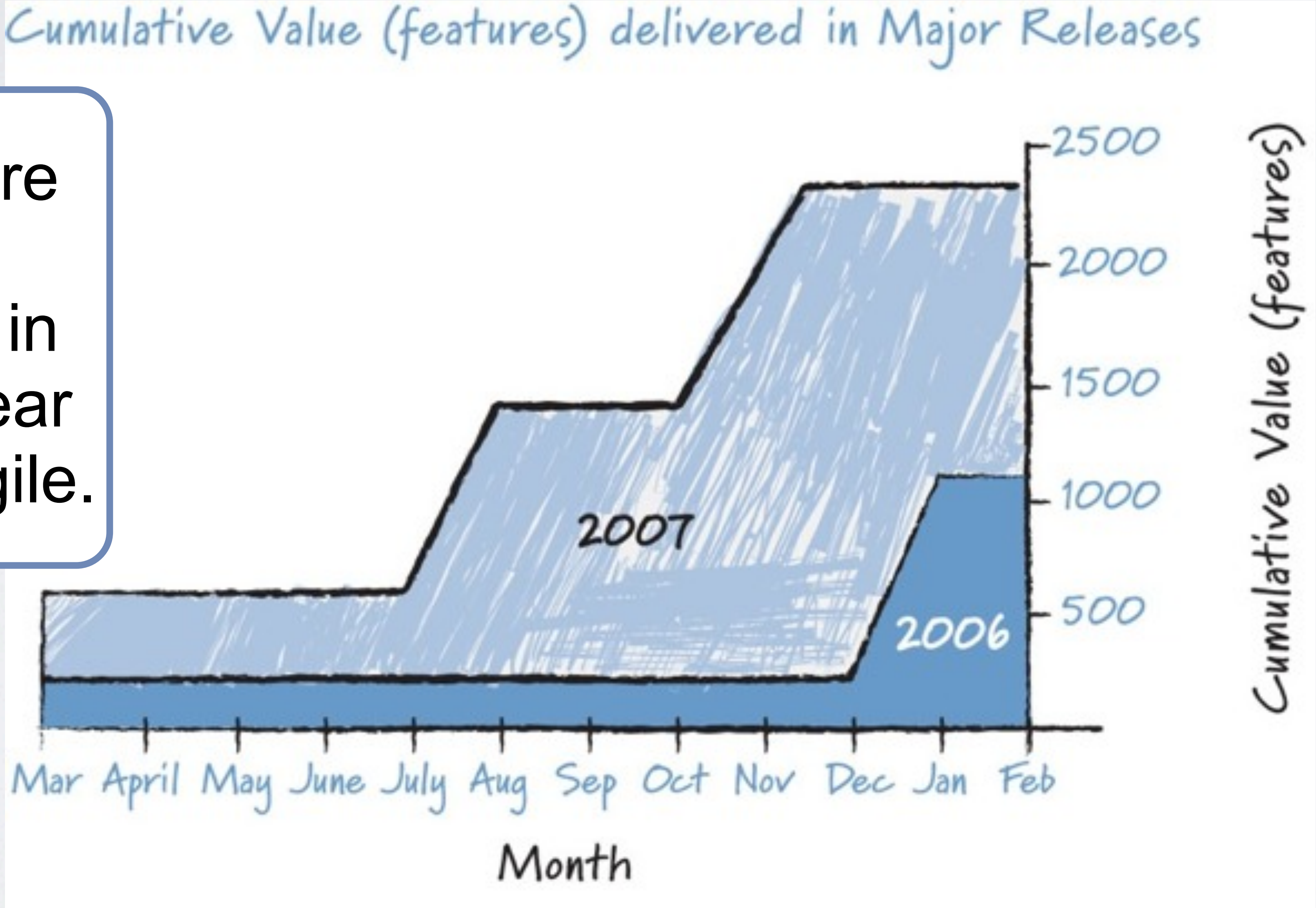
- VersionOne - "The State of Agile Development" 2011 (6,042 responses)



DOES AGILE WORK?

Salesforce.com

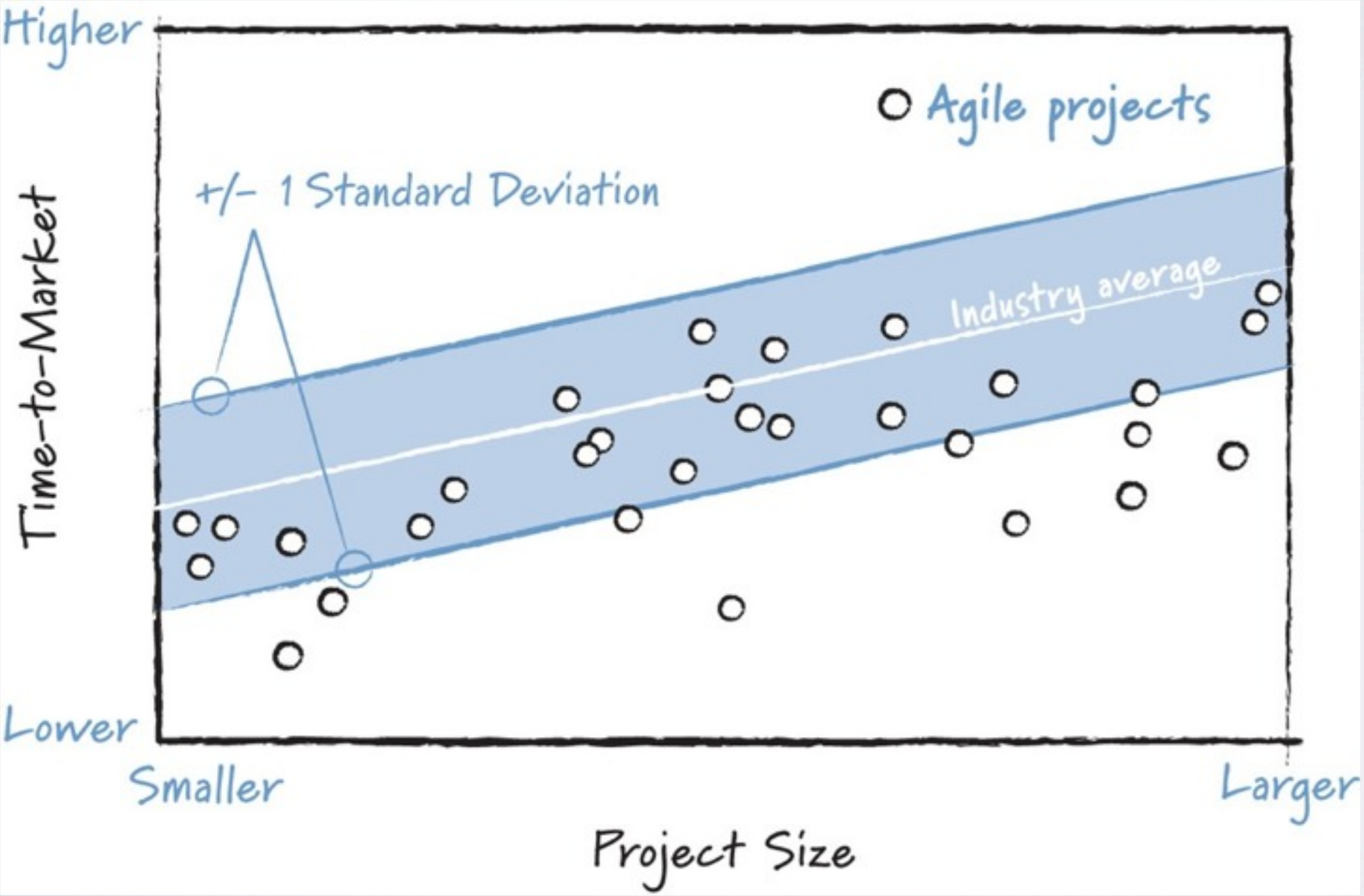
568% more value delivered in the first year of being agile.



Source: Salesforce.com, Green & Fry & Mountain Goat Software

DOES AGILE WORK?

Faster time to market



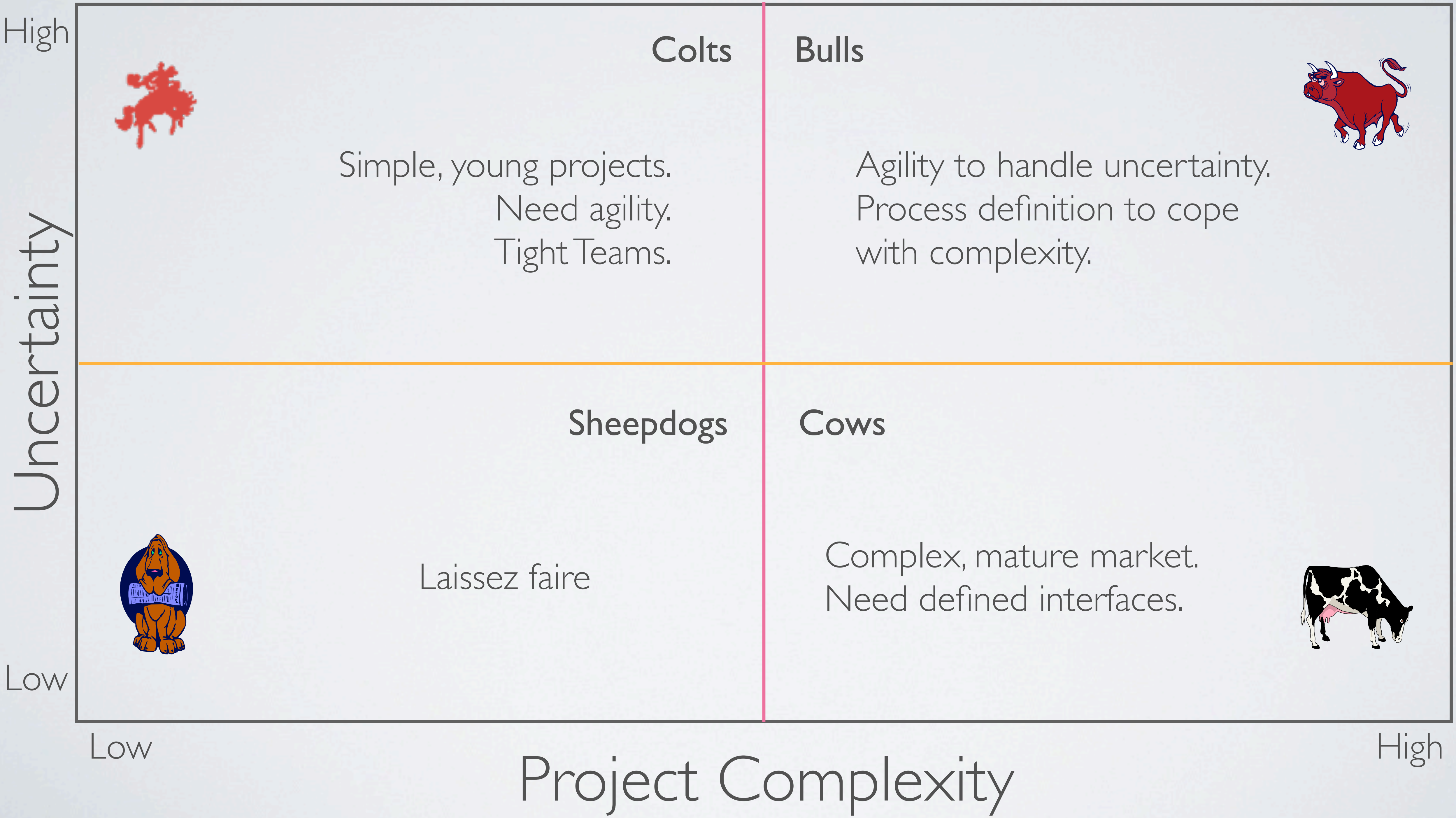
Agile projects are 16% more productive at a statistically significant level of confidence.

Source: Mah 2008 & Mountain Goat Software

DOWNSIDES OF AGILE

- Releases may be to often for customers to handle (enterprise software)
- Technical debt
- Documentation can be lacking
- Hard to scale properly, may require colocation

SHOULD YOU USE AGILE?



Uncertainty

High

Low

Colts

Simple, young projects.
Need agility.
Tight Teams.

Bulls

Agility to handle uncertainty.
Process definition to cope with complexity.

Sheepdogs

Laissez faire

Cows

Complex, mature market.
Need defined interfaces.

Low

High

Project Complexity

SHOULD YOU USE AGILE?

- Better than chaos
- Easier to transition to a lightweight method than heavyweight
- Easier to follow with less bureaucracy
- Co-operative team
- Strong business leadership
- Requires alignment with company philosophy and culture

WHEN NOT TO USE AGILE

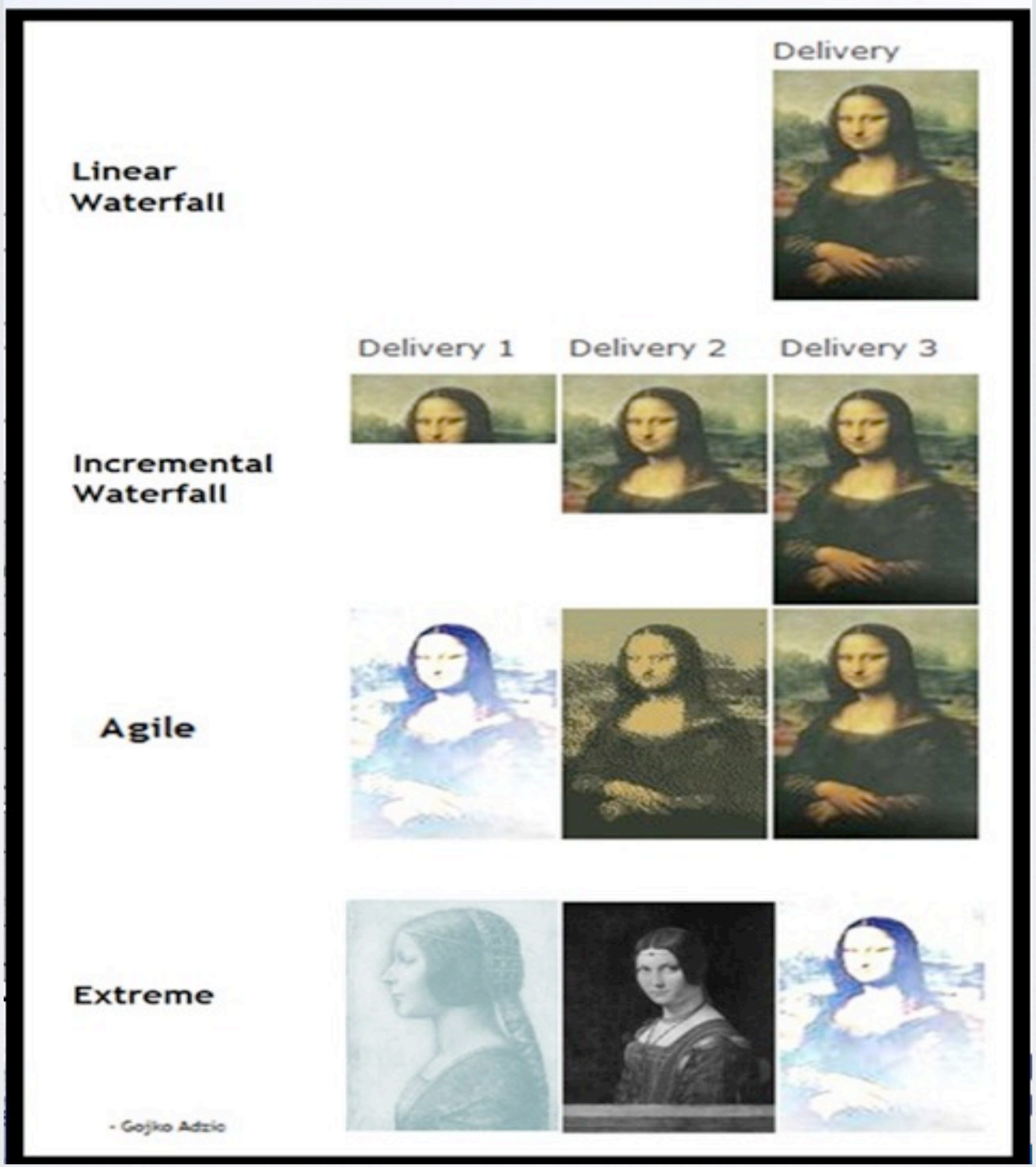
- Heavyweight methodologies can be successful when:
  - Requirements rarely change
  - The technology is mature
  - New and unknown is minimal
  - It’s been done before
- How many projects fit that description?

AGILE METHODOLOGIES

- Crystal Methods (Crystal Clear, Yellow, Orange and Red)
- Feature Driven Development (FDD)
- Extreme Programming (XP)
- Dynamic Systems Development Method (DSDM)
- Scrum
- Adaptive Project Framework (APF)

ADAPTIVE SOFTWARE DEVELOPMENT

AGILE


Delivery
Linear Waterfall
Delivery 1
Delivery 2
Delivery 3
Incremental Waterfall
Agile
Extreme
- Gojko Adzic

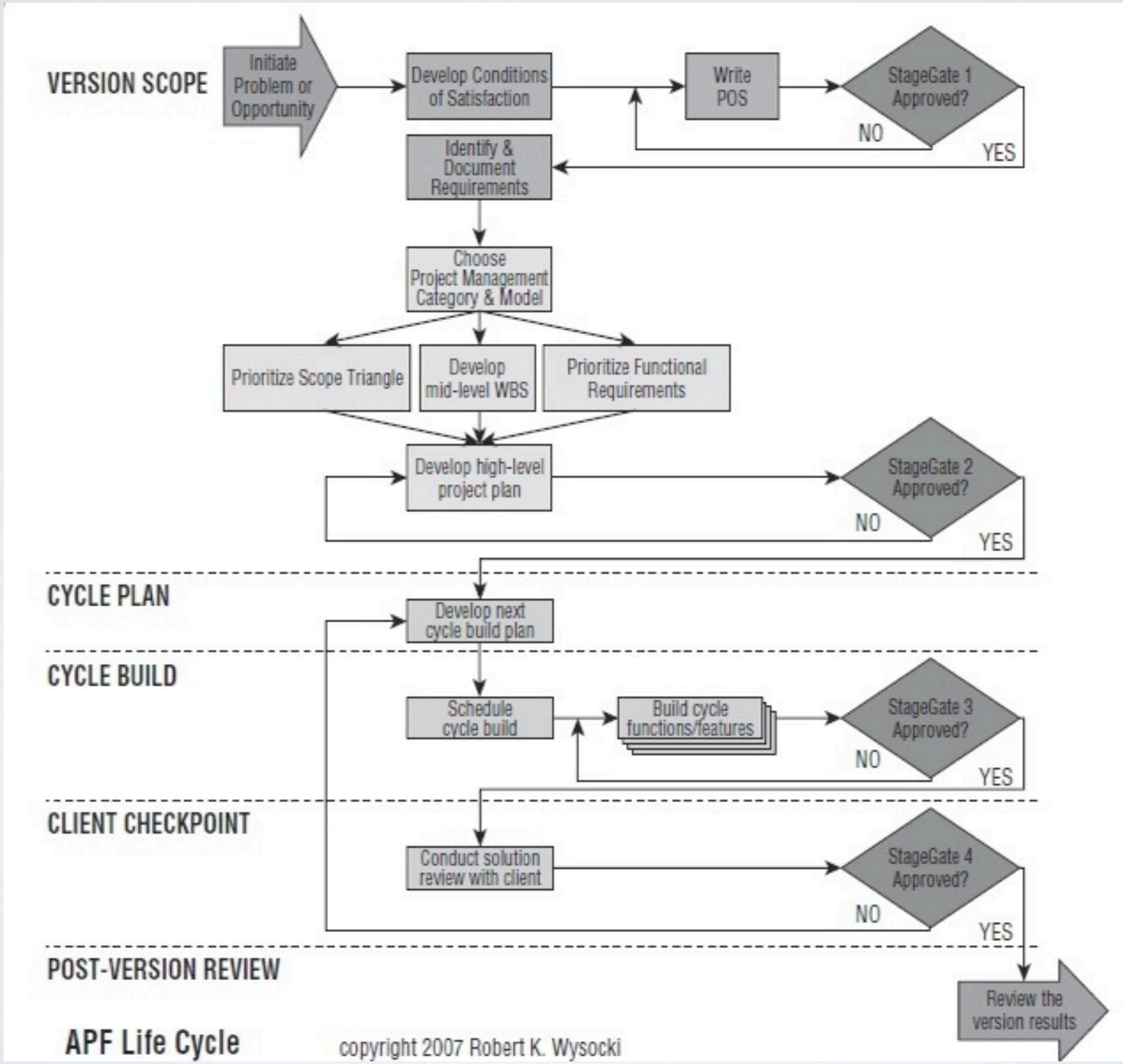
ADAPTIVE PROJECT FRAMEWORK (APF)

- General approach
- Fundamental concept scope is variable within time and cost constraints
- Maximizes business value
- Client focused and Client driven
- Change is embraced and not avoided
- Just in time planning
- Principle that you learn by doing

APF CORE VALUES

- Client-focused
- Client-driven
- Incremental results early and often
- Continuous questioning and introspection
- Change is progress to a better solution
- Don’t speculate on the future
- Emphasizes “learning”
- Iterative
- Time-boxed
- Risk driven and change-tolerant

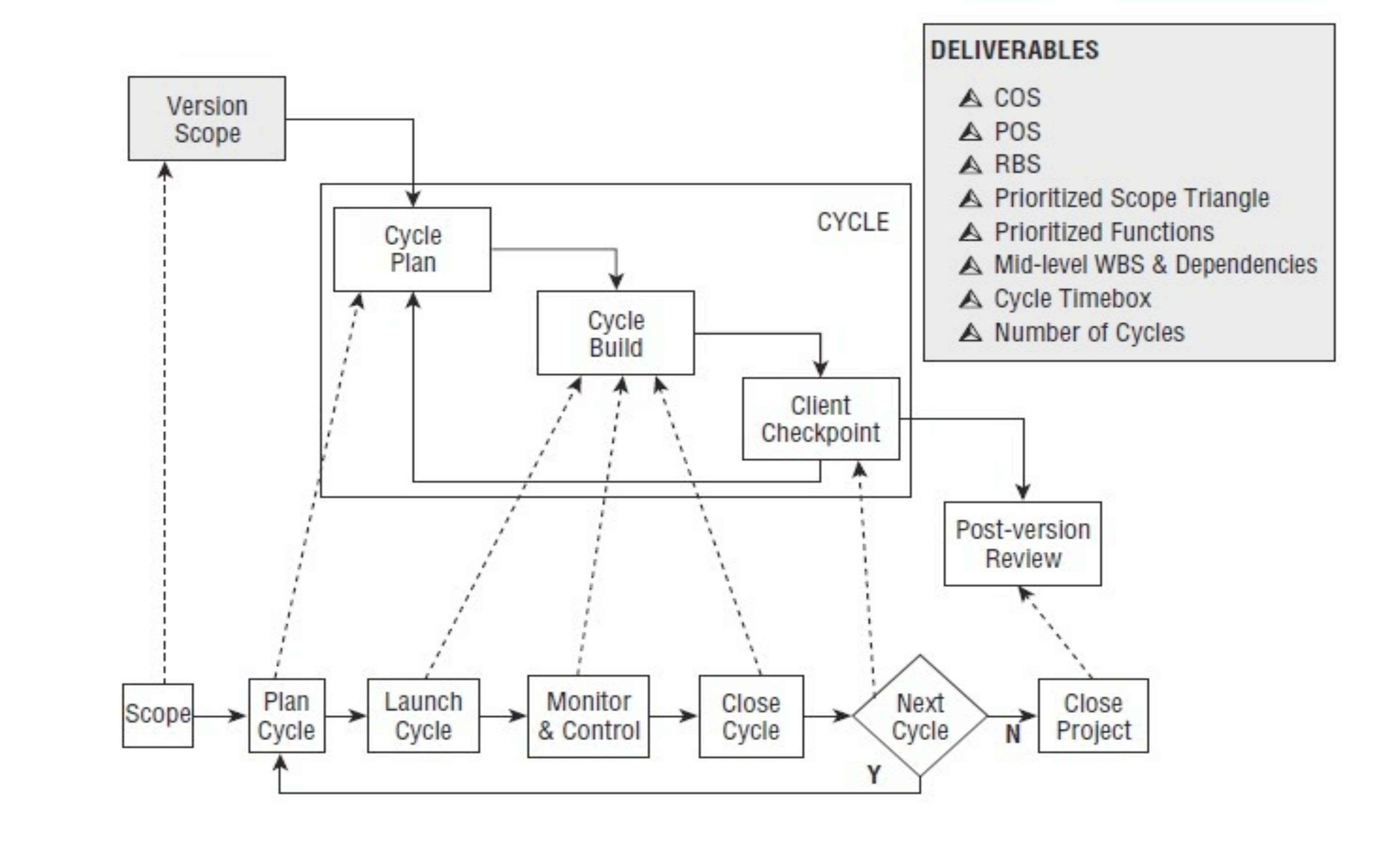
APF LIFE CYCLE


VERSION SCOPE
Initiate Problem or Opportunity
Develop Conditions of Satisfaction
Write POS
StageGate 1 Approved?
NO
YES
Identify & Document Requirements
Choose Project Management Category & Model
Prioritize Scope Triangle
Develop mid-level WBS
Prioritize Functional Requirements
Develop high-level project plan
StageGate 2 Approved?
NO
YES
CYCLE PLAN
Develop next cycle build plan
CYCLE BUILD
Schedule cycle build
Build cycle functions/features
StageGate 3 Approved?
NO
YES
CLIENT CHECKPOINT
Conduct solution review with client
StageGate 4 Approved?
NO
YES
POST-VERSION REVIEW
Review the version results

APF Life Cycle

copyright 2007 Robert K. Wysocki

APF – VERSION SCOPE



DELIVERABLES

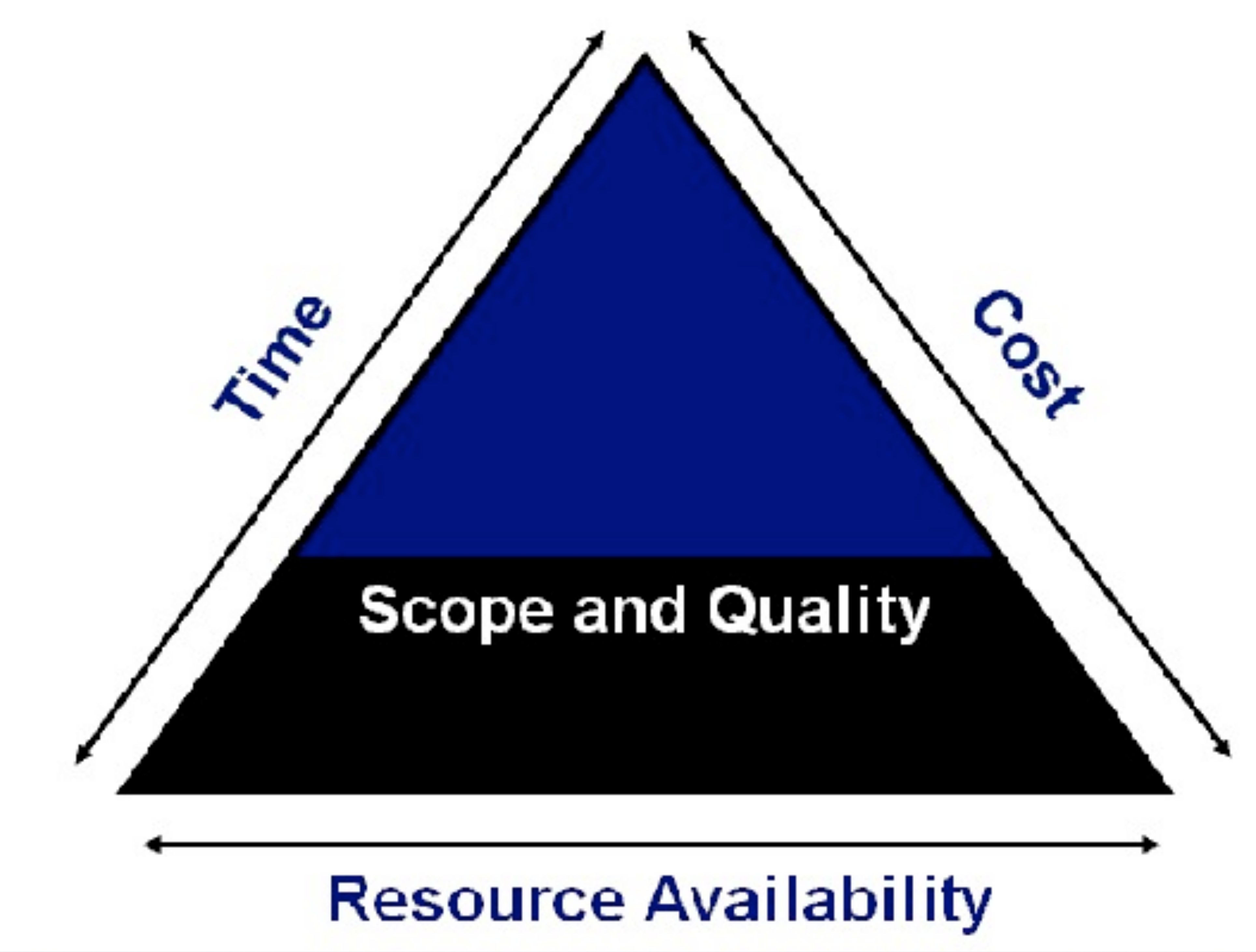
- COS
- POS
- RBS
- Prioritized Scope Triangle
- Prioritized Functions
- Mid-level WBS & Dependencies
- Cycle Timebox
- Number of Cycles

VERSION SCOPE

- COS is input to decision on what PMLC model to be used and then write POS.

- Five components of scope triangle are prioritized for decision making and problem solving in cycle build phase

APF – SCOPE TRIANGLE



PRIORITIZATION APPROACHES FOR SCOPE TRIANGLE

- Forced Ranking
- Paired Comparison
- MoSCoW

PRIORITIZATION APPROACHES – FORCED RANKING

MANAGER FUNCTION	A	B	C	D	RANK SUM	FORCED RANK
1	2	3	2	4	11	3
2	4	1	1	2	8	1
3	6	2	5	5	18	5
4	1	5	3	1	10	2
5	3	4	4	3	14	4
6	5	6	6	6	23	6

PRIORITIZATION APPROACHES – PAIRED COMPARISON

	1	2	3	4	5	6	SUM	RANK
1	X	1	1	0	1	1	4	2
2	0	X	1	0	1	1	3	3
3	0	0	X	0	0	1	1	5
4	1	1	1	X	1	1	5	1
5	**0**	0	1	0	X	1	2	4
6	0	0	0	0	0	X	0	6

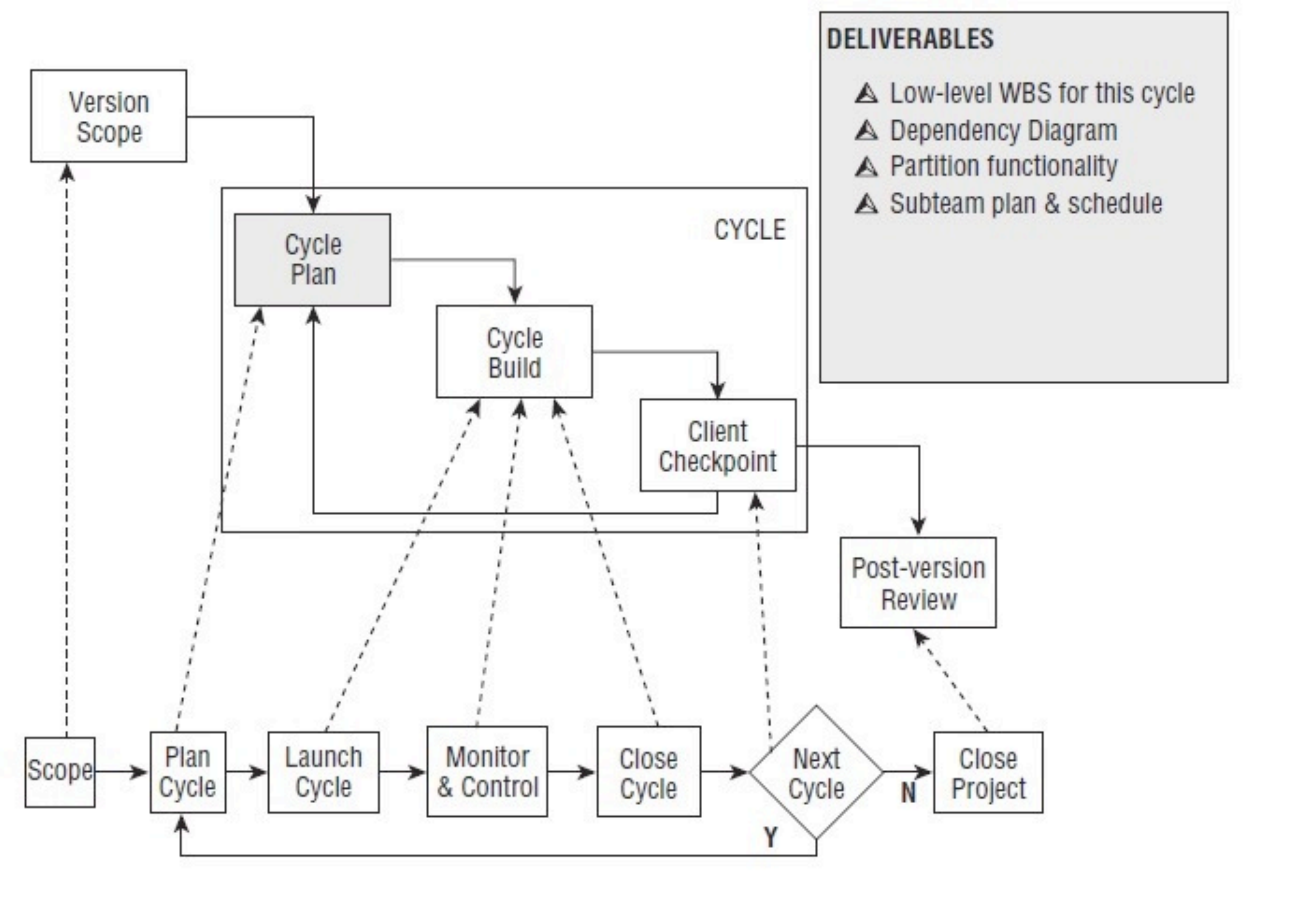
PRIORITIZATION APPROACHES - MOSCOW

M:	Must Have
S:	Should Have
C:	Could Have
W:	Would be Nice to Have

APF – SCOPE TRIANGLE RANKING

Priority	Critical (1)	(2)	(3)	(4)	Flexible (5)
Scope				✗	
Quality			✗		
Time	✗				
Cost					✗
Resource Availability		✗			

APF – CYCLE PLAN



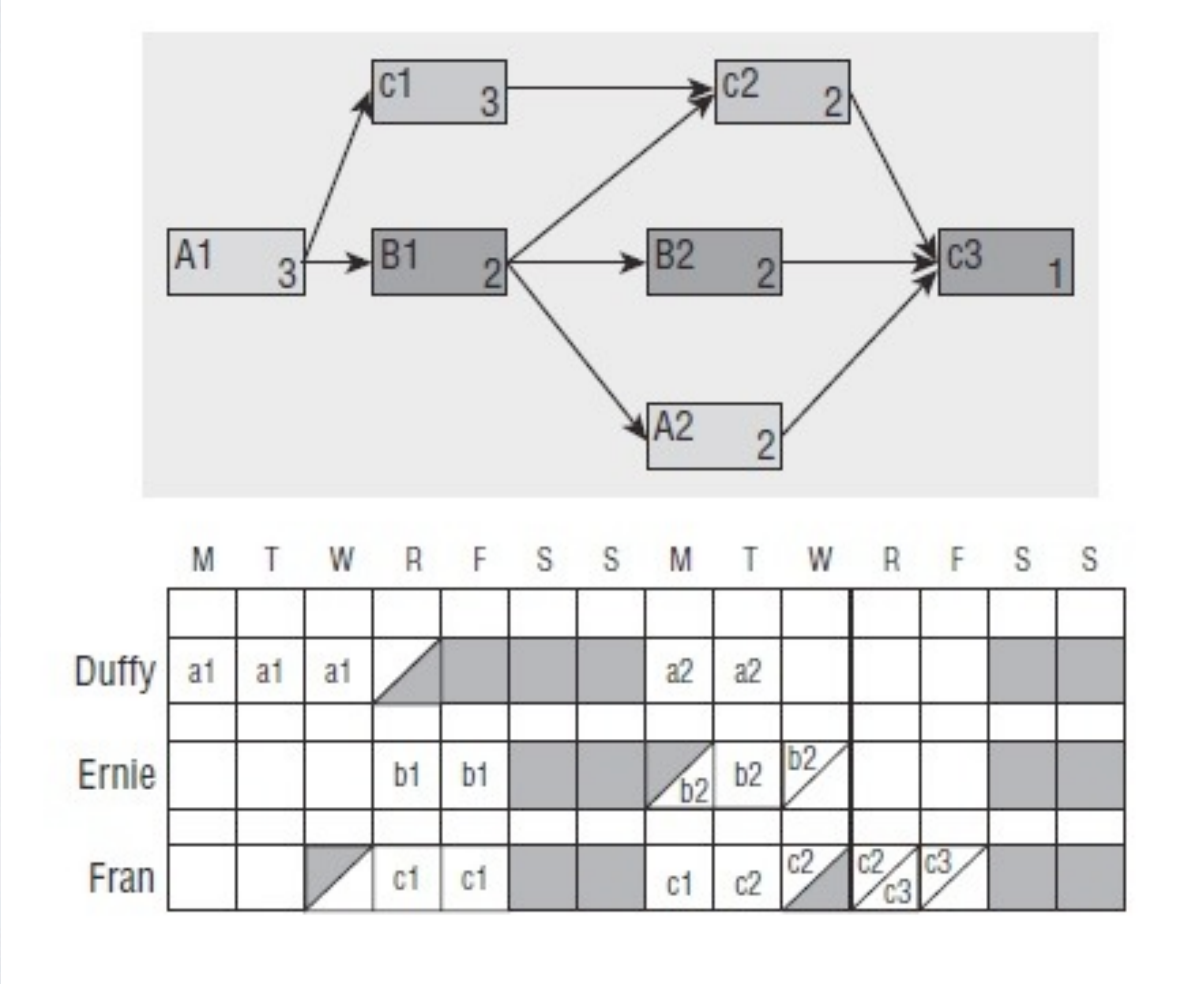
DELIVERABLES

- Low-level WBS for this cycle
- Dependency Diagram
- Partition functionality
- Subteam plan & schedule

APF CYCLE PLANNING EFFORT

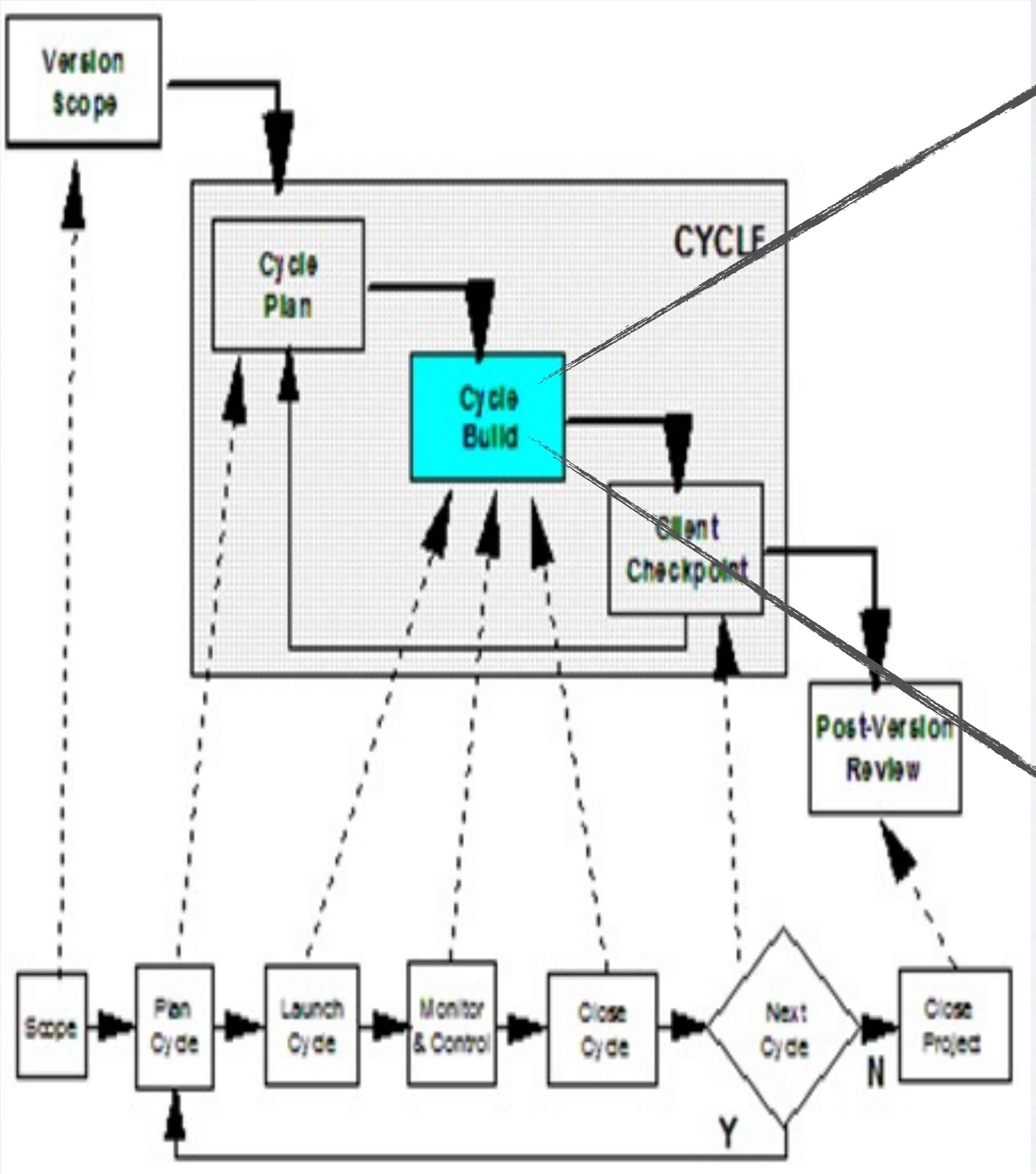
- Extract from the WBS the functions to be built
- Complete the extracted WBS down to the task level
- Build the dependency network diagram
- Partition the tasks into independent meaningful groups and assign teams to each group
- Each team will develop a plan and schedule

APF – RESOURCE LOADED SCHEDULE



	M	T	W	R	F	S	S	M	T	W	R	F	S	S
Duffy	a1	a1	a1	▨	▨	▨	▨	a2	a2				▨	▨
Ernie				b1	b1	▨	▨	▨ b2	b2	b2			▨	▨
Fran			▨	c1	c1	▨	▨	c1	c2	c2 ▨	c2 c3	c3	▨	▨

CYCLE BUILD

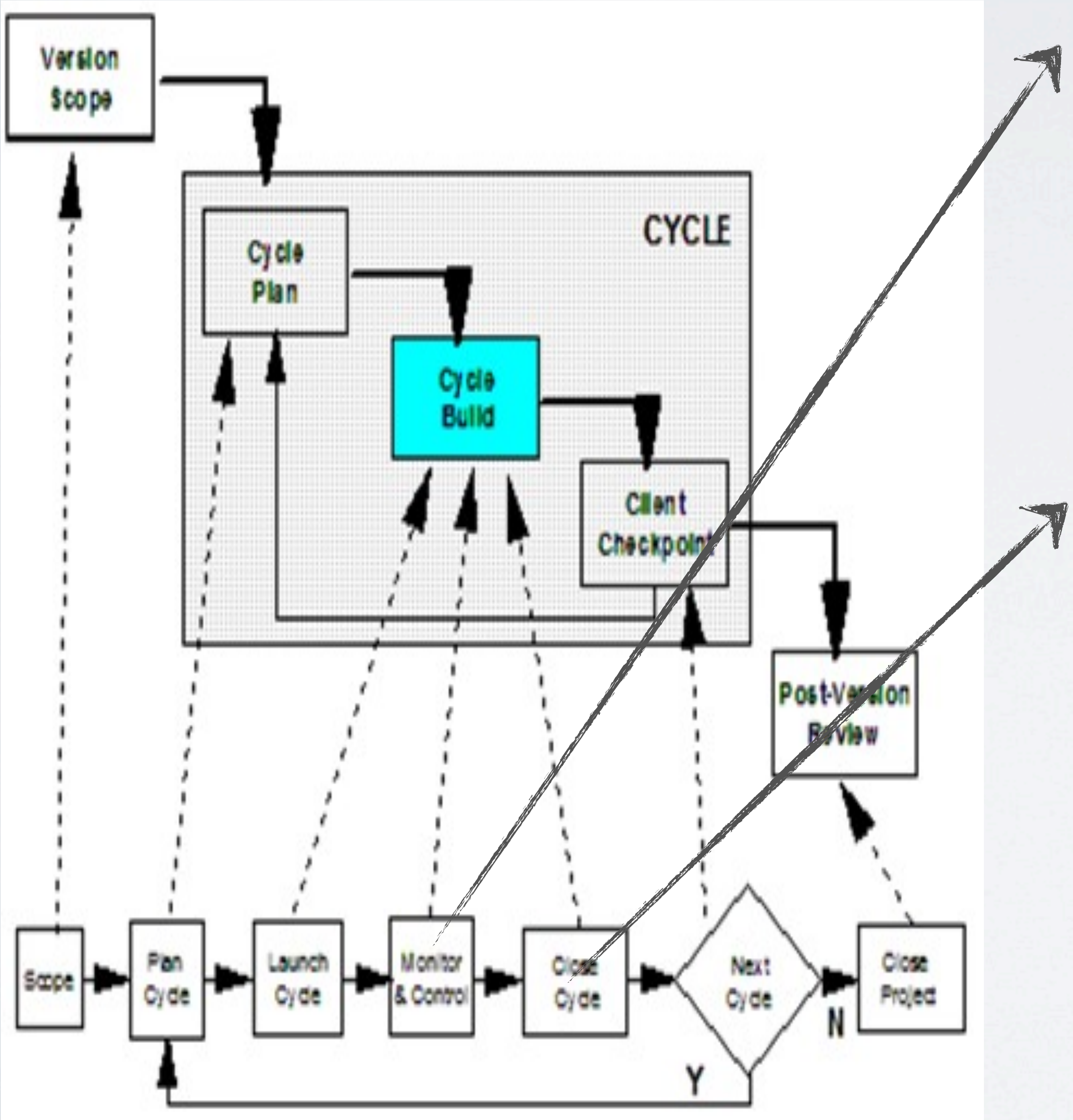


- Conduct detailed planning for producing the functionality assigned to this cycle
- Begin cycle work
- Monitor and adjust cycle build
- Create a Scope Bank
- Create Issues
- Build Cycle Functionality
- Monitoring and Control

Measuring Team Strength:

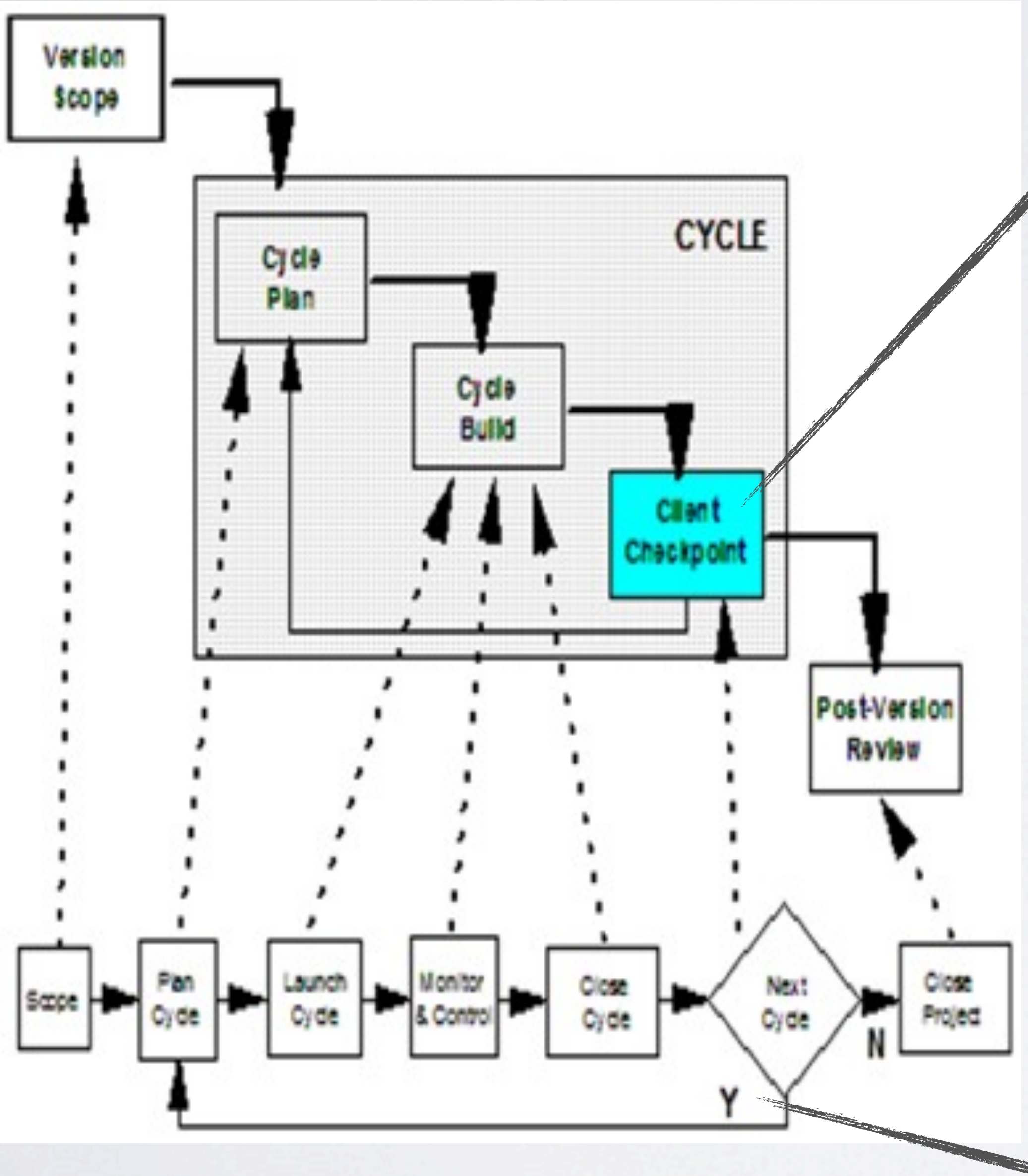
TS = # of We’s / # of I’s + # We’s

CYCLE BUILD



- Monitor and Control:
  - Team status meeting
  - Major issues are posted in Issue Log
- Close Cycle:
  - Time Box expired
  - All swim lanes competed early
  - A major problem occurs

CLIENT CHECKPOINT

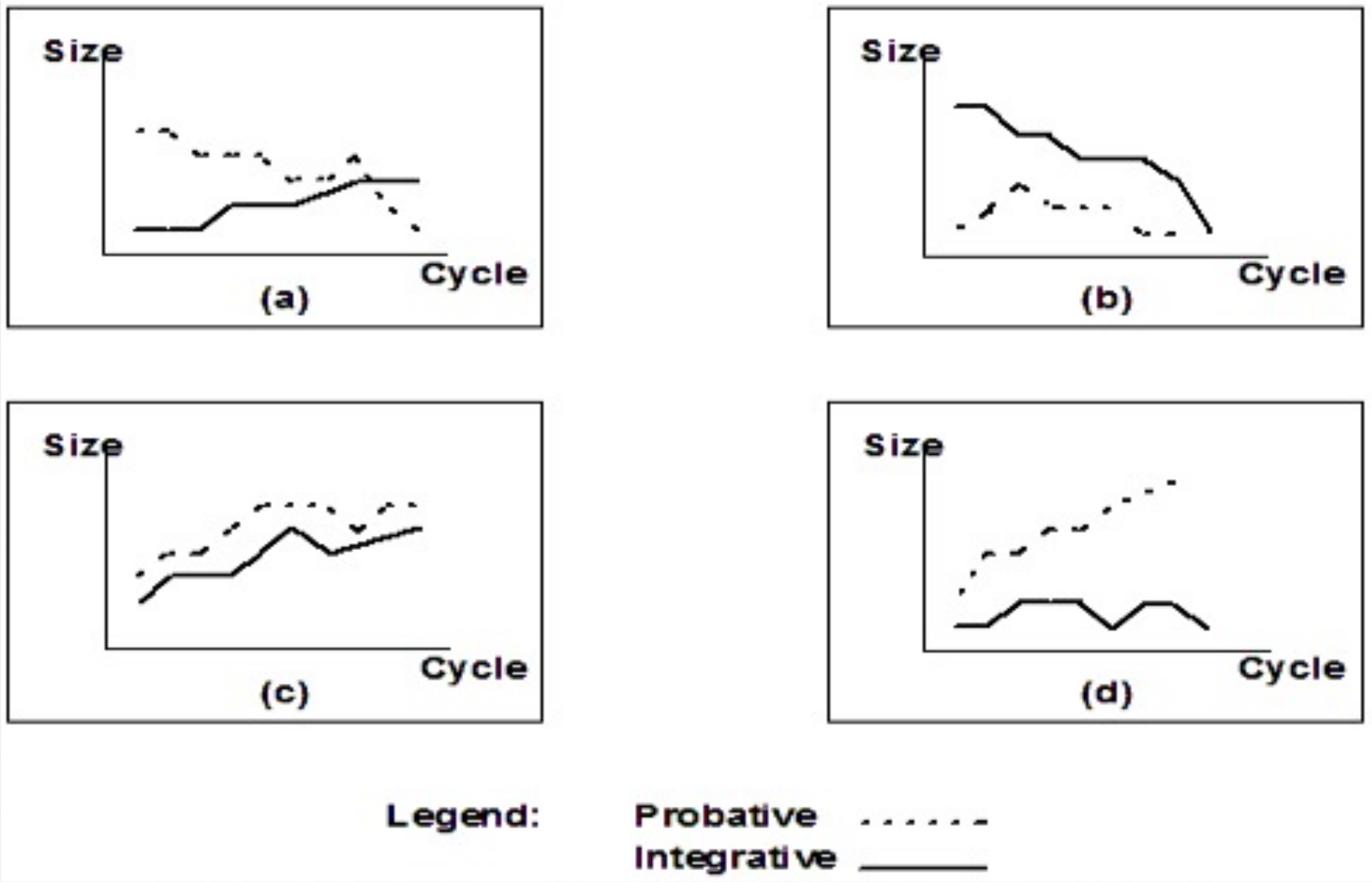


- Client and project team perform a quality review of the functionality produced
- The sequence Cycle Plan / Cycle Build / Client Checkpoint is repeated
- Operate as Co-project manager

If the client accepts everything, the team will move to the next cycle.

SCOPE BANK

- Identify and prioritize Probative Swim Lane
- Identify and prioritize Integrative Swim Lane

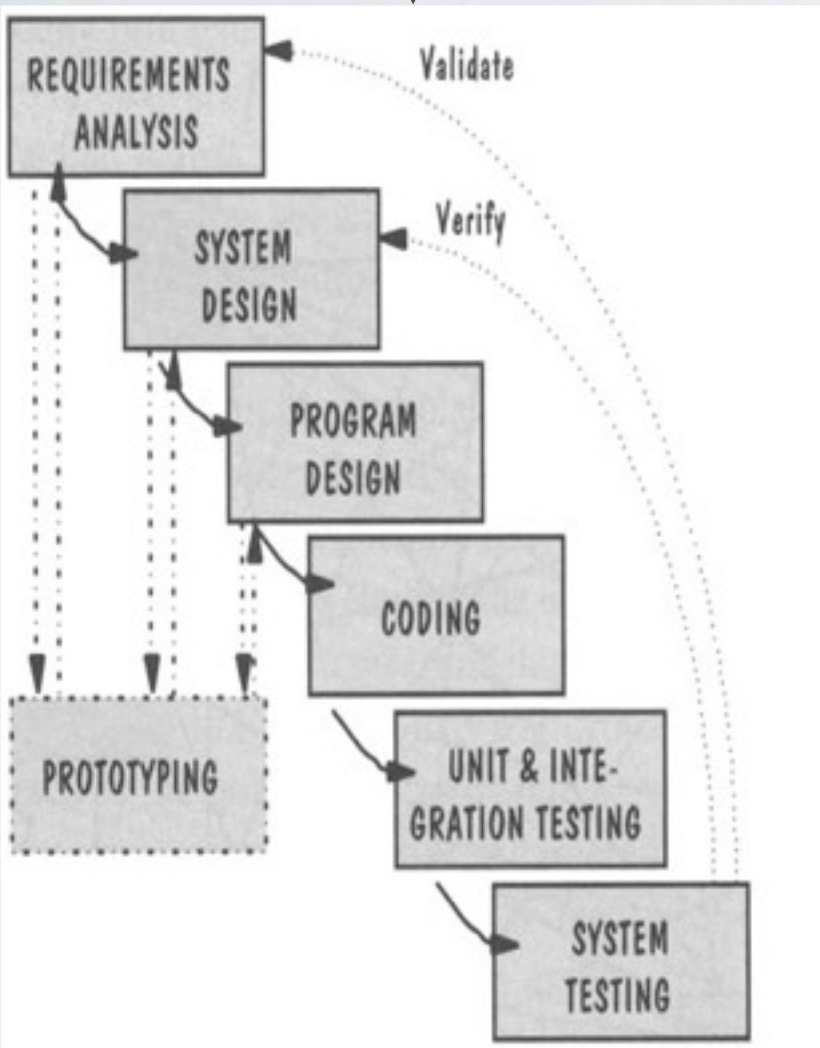


POST-VERSION REVIEW

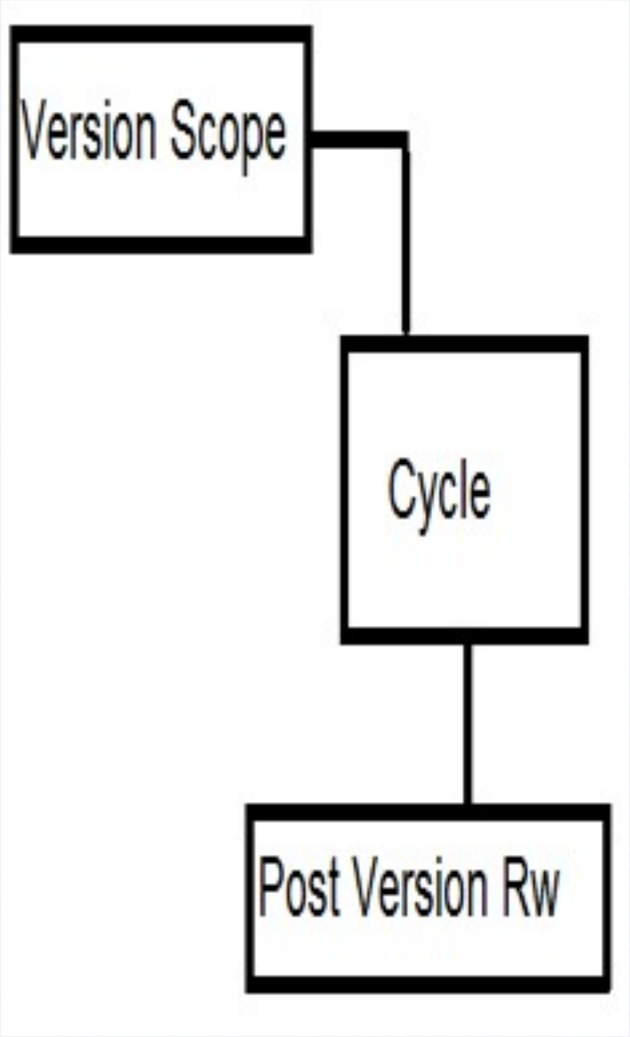
- Determine if the expected business outcome was realized
- Determine what was learned that can be used to improve the solution
- Determine what was learned that can be used to improve the effectiveness of APF

ADAPTING APF

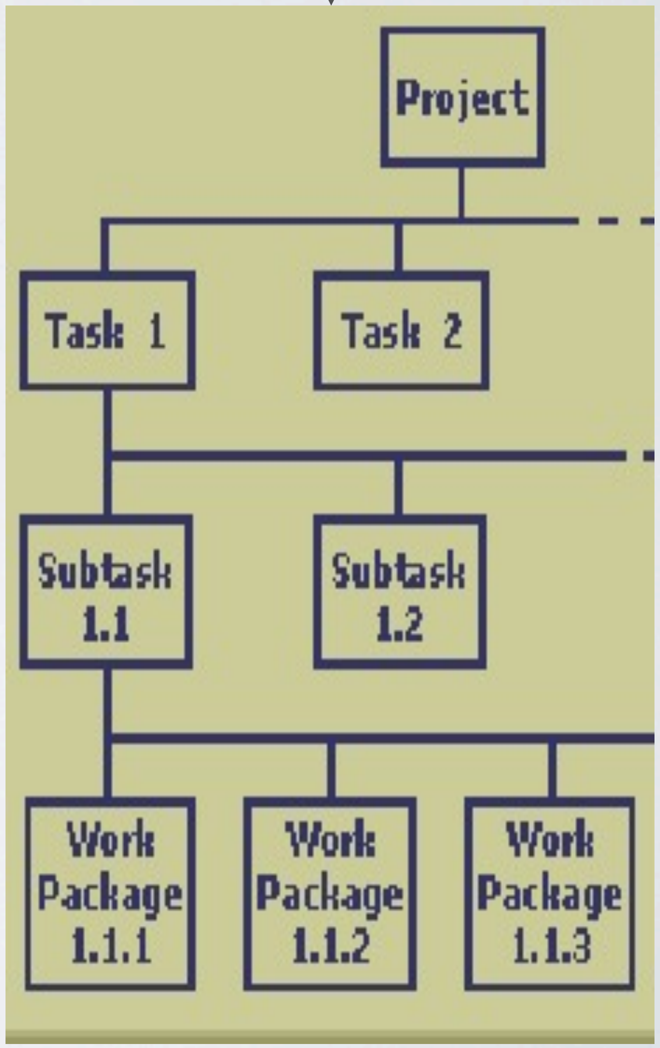
Proof of Concept Cycle



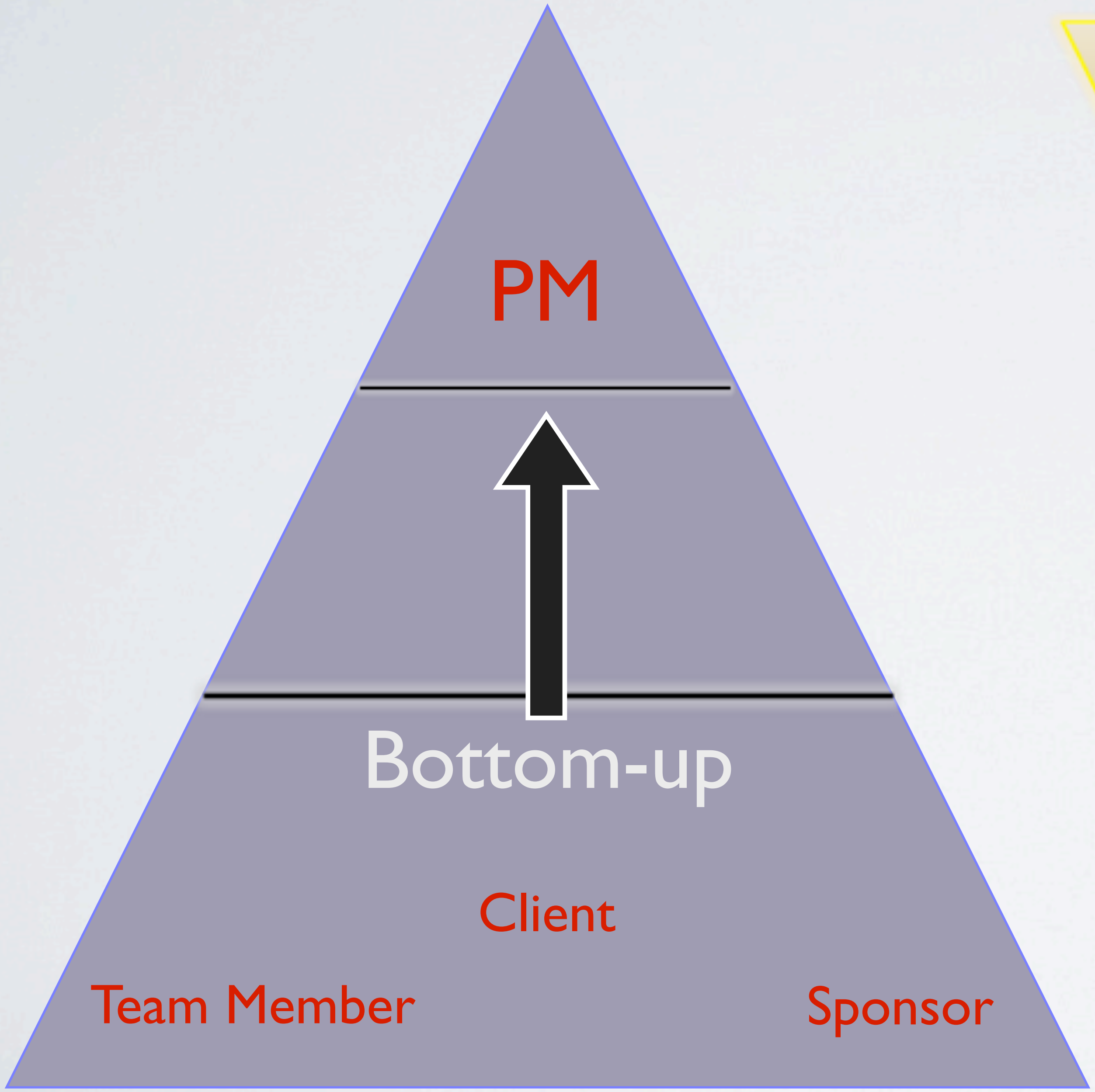
Revising the Version Plan

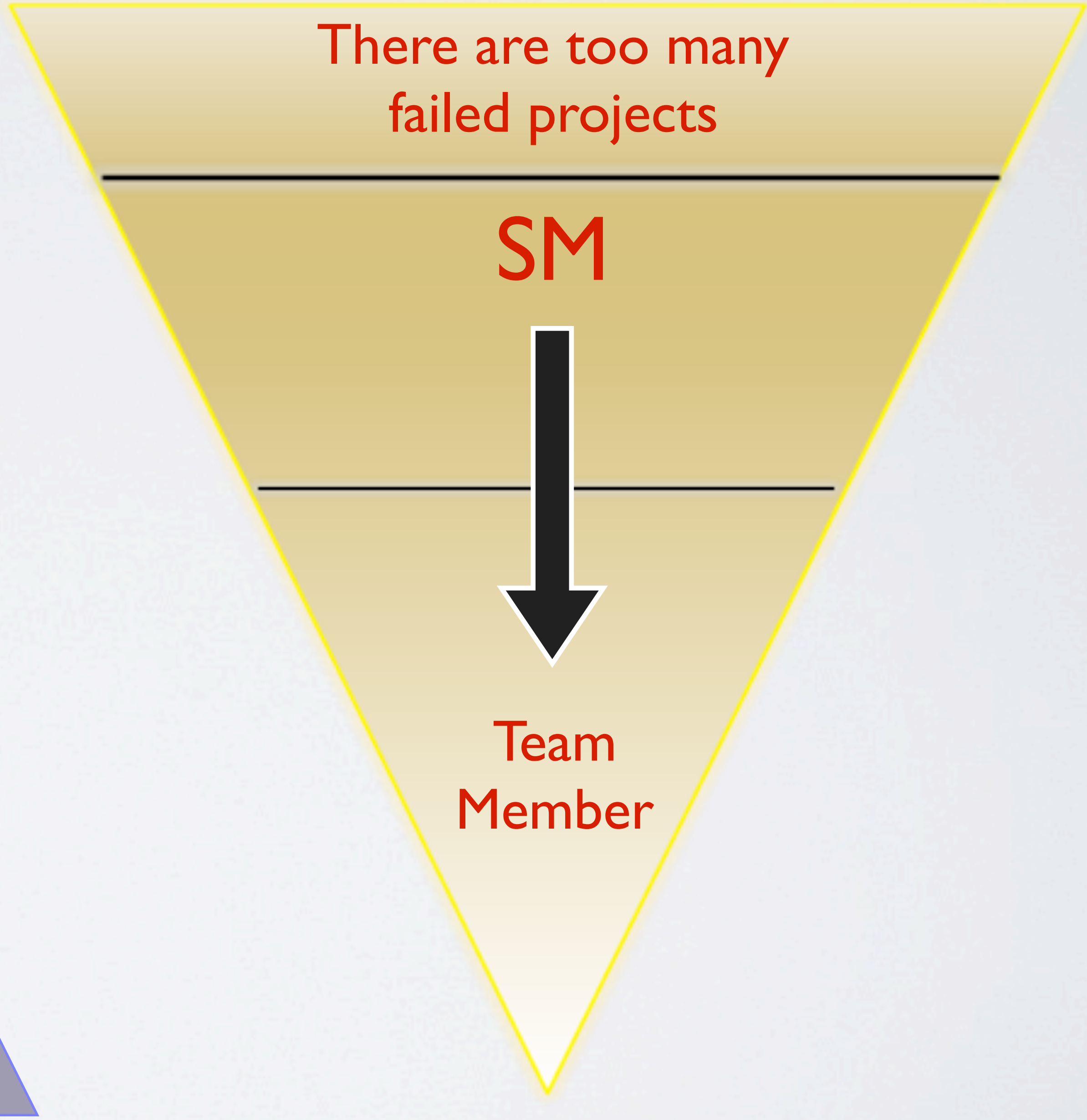


Embedding APF



IMPLEMENTING APF





EXTREME PROGRAMMING

WHAT IS EXTREME PROGRAMMING (XP)?

- A **software development methodology** which is intended to improve software quality and responsiveness to changing customer requirements.

- As a type of agile software development, it advocates frequent "**releases**" in short development cycles.

- Help improve productivity and introduce checkpoints where new **customer requirements** can be adopted.

HISTORY

- Extreme programming was created by **Kent Beck** during his work on the **Chrysler Comprehensive Compensation System** (C3) payroll project.

- Beck became the C3 **project leader** in March 1996 and began to refine the development method used in the project and wrote a book on the method (in October 1999, Extreme Programming Explained was published)

GOAL OF XP

XP attempts to reduce the cost of changes in requirements by having multiple short development cycles, rather than a long one.

XP - PROCESS

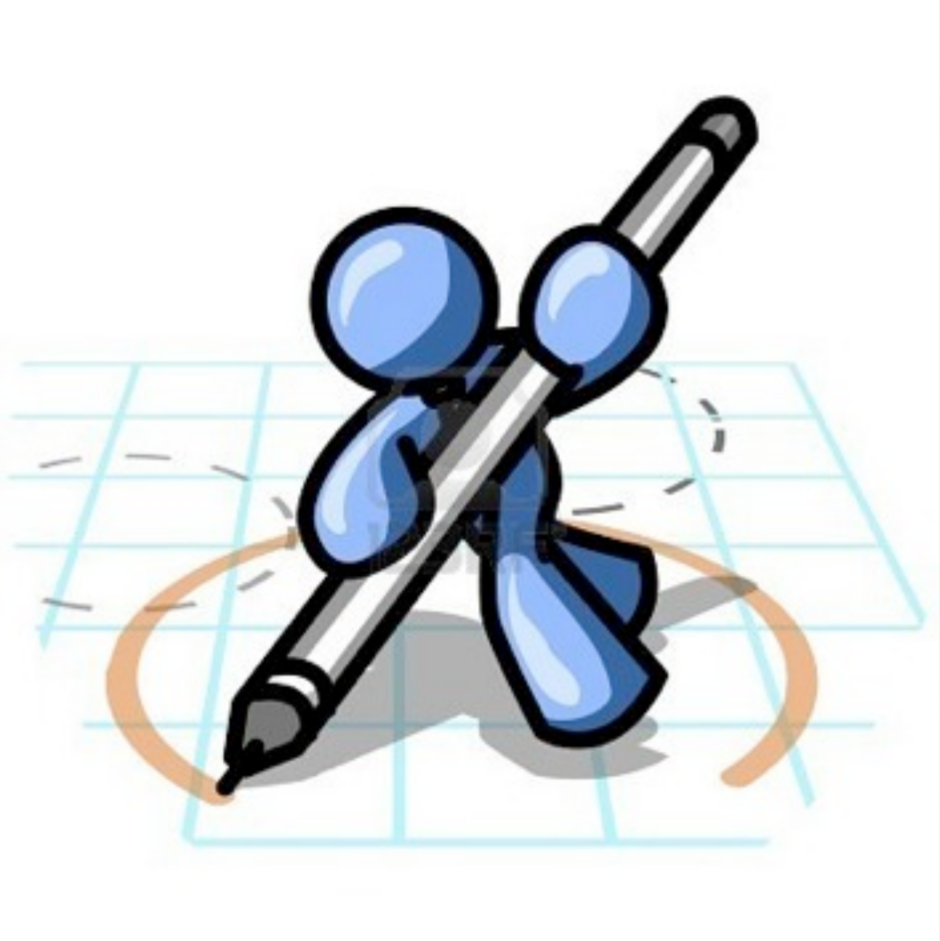
- Planning
- Designing
- Coding
- Testing

XP - PLANNING

- User Stories
  - Functionalities of the system are described using stories, short descriptions of customer-visible functionalities
- Small & Short Releases
  - Every release should be as small as possible, containing the most valuable business requirements
  - It is far better to plan a month or two at a time than six months or a year at a time

XP - DESIGNING

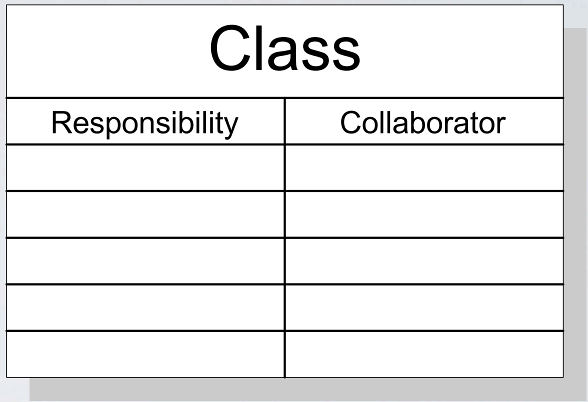
- Simple Design (KISS)
- CRC Cards: Class, Responsibility, Collaborator
- Spike Solutions
- Refactoring



SIMPLE DESIGN

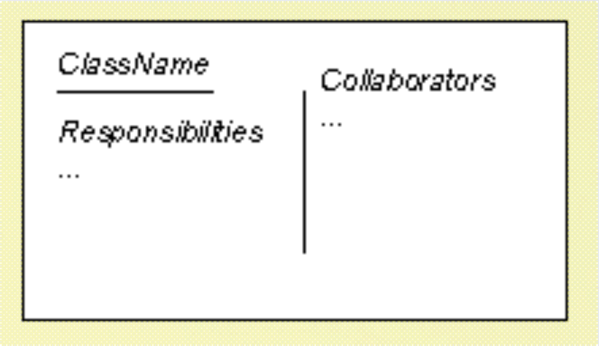
- Misconception about XP: XP advises to avoid design
- Truth: XP advises:
  - Avoid too much Up Front Design / extra design at early phase, as requirement is not clear.
  - Simple and elegant design
  - Avoid over design

CRC CARDS



Class

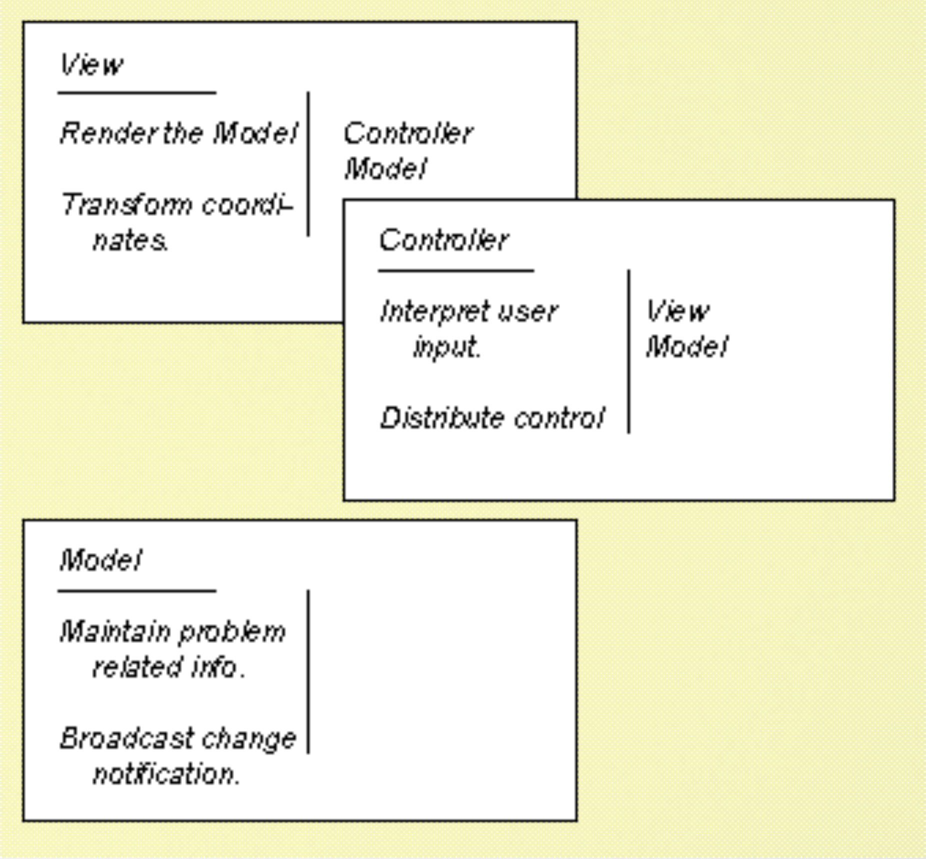
Responsibility	Collaborator



ClassName

Responsibilities
...

Collaborators
...



View

Render the Model

Transform coordinates.

Controller
Model

Controller

Interpret user input.

Distribute control

View
Model

Model

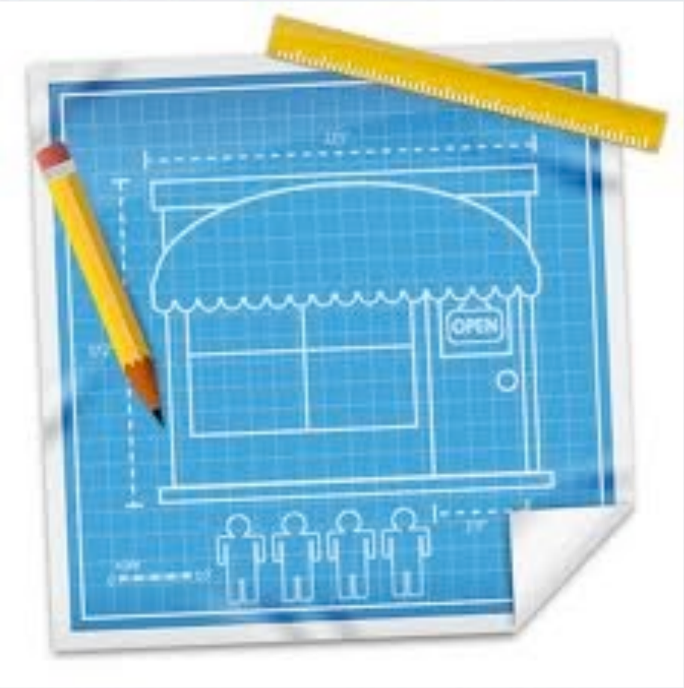
Maintain problem related info.

Broadcast change notification.

- Used to identify Classes, Responsibilities, & Collaborations between objects
- Created from index cards with with this information:
  - Class name
  - Responsibilities of the class
  - Names of other classes with which the class will collaborate to fulfill its responsibilities.
  - Author

SPIKE SOLUTION

1. A design problem occurs
2. Create a **PROTOTYPE** of that portion of the design
3. Implement and evaluate the prototype



Intent:
To lower the risk when true implementation starts.

REFACTORING

Changing a software system in such a way that:

1. The internal structure is improved
2. The external behavior is not altered (not changed)

Also means:

Design occurs **CONTINUOUSLY** as the system is constructed

XP - CODING

COMMON PRACTICES

- Code the **Unit test** first
- Pair Programming
- Continuous Integration (CI)
- Leave **Optimization** until last

PAIR PROGRAMMING

- Two people work together at one computer to create code for a story

- This provides a mechanism for real-time problem solving and real-time quality assurance

- Keeps the developers focused on the problem at hand

- As pair programmers complete their work, the code they develop is integrated with the work of others

CONTINUOUS INTEGRATION

- Avoidance of "big bang" integrations
- Integration for each pair occurs several times each day
- All tests run prior to commitment to code base
- Makes the cause of failures more obvious
- Minimizes merge pain
- Facilitates the code base evolving steadily
- Forces bug fixing to occur immediately
- Often supplemented by daily builds
- Contributes to the avoidance of quality & integration debt

CODING STANDARDS

- Consensus of coding style and practices
- Facilitates moving about the code base
- Contributes to definition of clean code and “doneness”
- Removes distraction of endless arguments
- Goal is that code looks anonymous
- Standards evolve over time

XP - TESTING

TESTING PRACTICES

- All code should have Unit Tests
- All code must pass all unit tests before it can be released
- When a bug is found tests are created
- Unit Tests and Acceptance tests are run often and the score is published

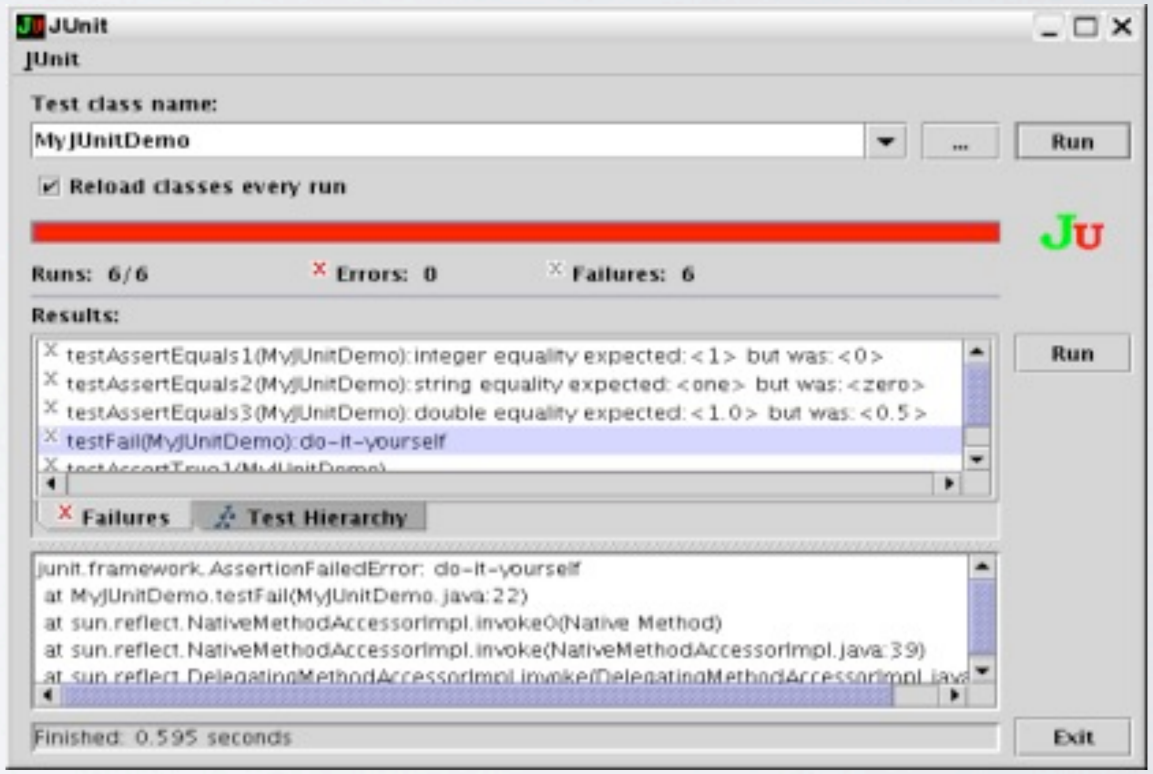
UNIT TESTS

- Unit tests are written **before** the code

- Tests are run to ensure that the software **fails**

- A **good** test case is one that ensures that the software fails

- Test is rerun **until it passes**



Test class name: MyJUnitDemo

☑ Reload classes every run

Runs: 6/6 Errors: 0 Failures: 6

Results:

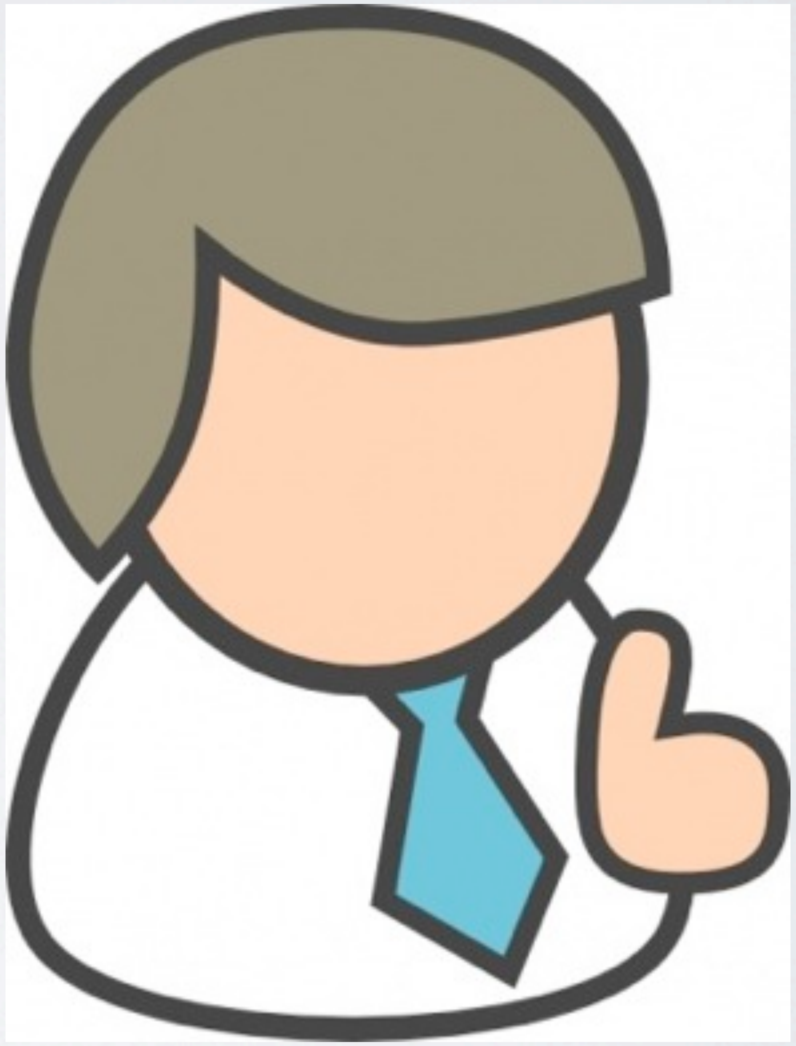
```
testAssertEquals1(MyJUnitDemo):integer equality expected:<1> but was:<0>
testAssertEquals2(MyJUnitDemo):string equality expected:<one> but was:<zero>
testAssertEquals3(MyJUnitDemo):double equality expected:<1.0> but was:<0.5>
testFail(MyJUnitDemo):do-it-yourself
testAssertTrue1(MyJUnitDemo)
```

```
junit.framework.AssertionFailedError: do-it-yourself
 at MyJUnitDemo.testFail(MyJUnitDemo.java:22)
 at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
 at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
 at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java
```

Finished: 0.595 seconds

ACCEPTANCE TEST

- Also known as customer tests
- They are specified by the customer
- They focus on the overall system features and functionality that are visible and reviewable by the customer
- They are derived from user stories

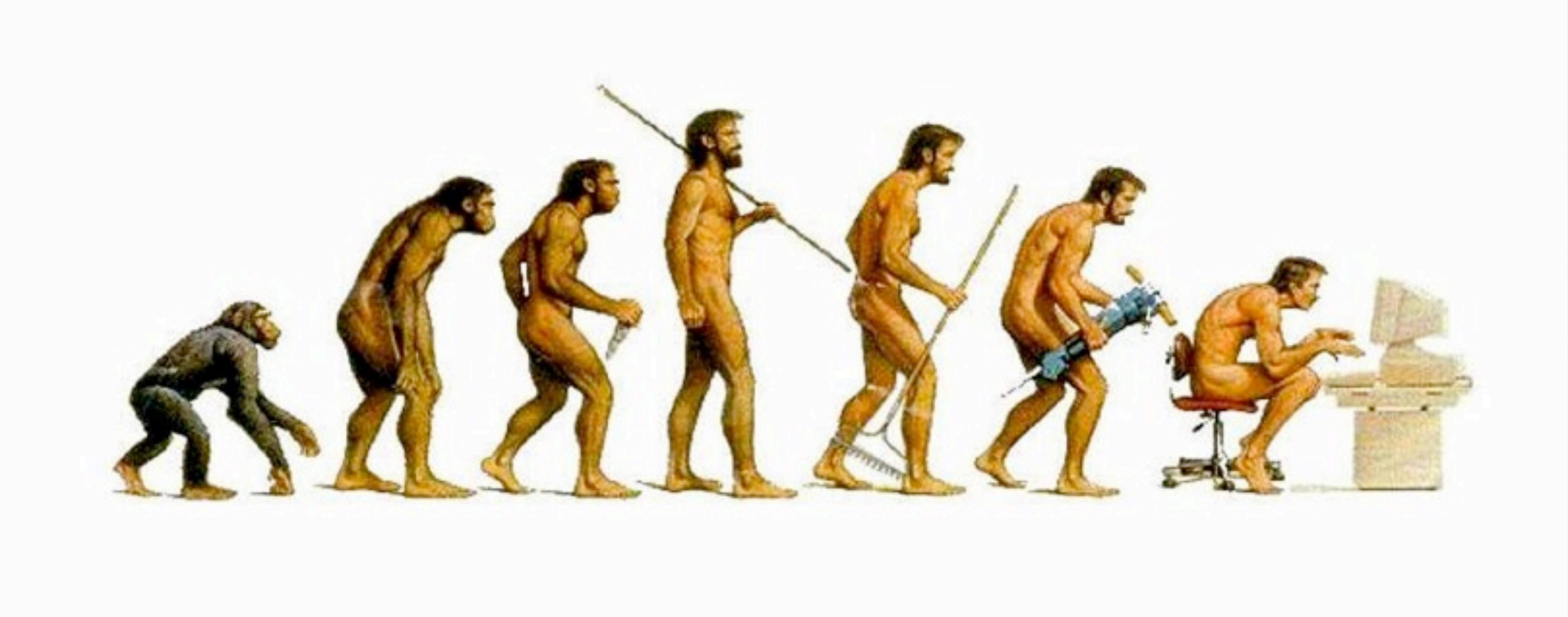


Hint:

Fixing small problems every few hours takes **less time** than fixing huge problems just before the deadline.

WHEN TO USE XP?

- Dynamically changing requirements
- Risky projects
- Small development groups (up to 100)
- Non-fixed price contract



DYNAMIC SYSTEMS DEVELOPMENT METHOD

INTRODUCTION TO DSDM

- An iterative and incremental approach that emphasis continuous user involvement
- DSDM became the number one framework for Rapid Application Development (RAD)
- Most mature agile development method
- DSDM is about people, not tools
- It seems ideally suited for software development that place a high importance on the user interface or usability aspects of products.

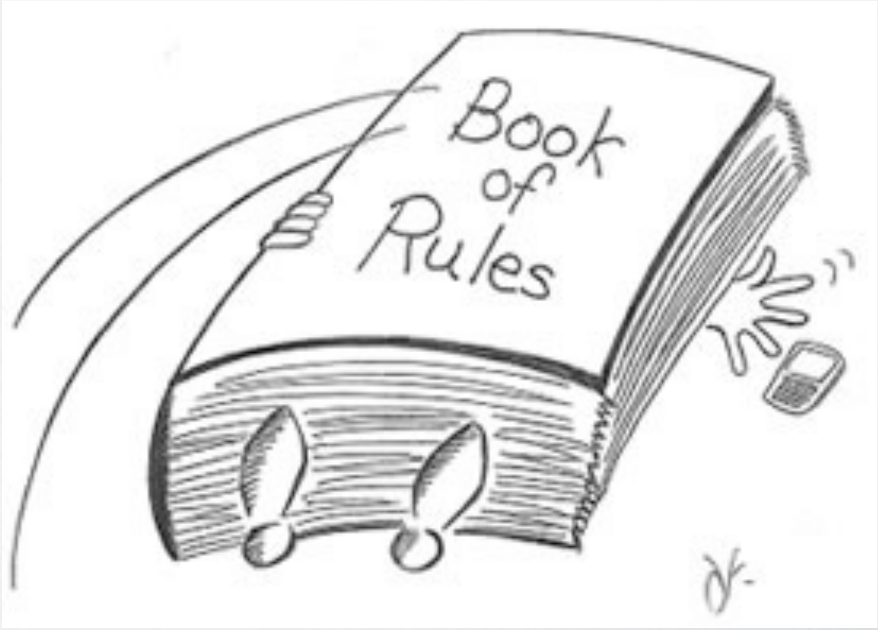
HISTORY OF DSDM

- Developed in the United Kingdom in the 1990's and was first released in 1995

- Developed by several vendors and experts in the field of Information Systems (IS), combining their best practice experiences

9 PRINCIPLES OF DSDM

1. Active user involvement is imperative
2. Team must be empowered to make decisions
3. Focus is on frequent delivery
4. Fitness for business is criterion for acceptance of deliverables
5. Iterative and incremental development is mandatory
6. All changes during development must be reversible
7. Requirements are base-lined at high-level
8. Testing is integrated throughout the cycle
9. Collaborative and Co-operative approach



PHASES OF DSDM

Pre-project

Includes project suggestion and selection of a proposed project candidate.

Feasibility study

Definition of the problem to be addressed, assessments of the likely cost and technical feasibility of delivering of a computer system to solve the business problem.

PHASES OF DSDM

Business Study

This stage examines the influenced business processes, user groups involved and their respective needs and wishes.

Functional Model Iteration (FMI)

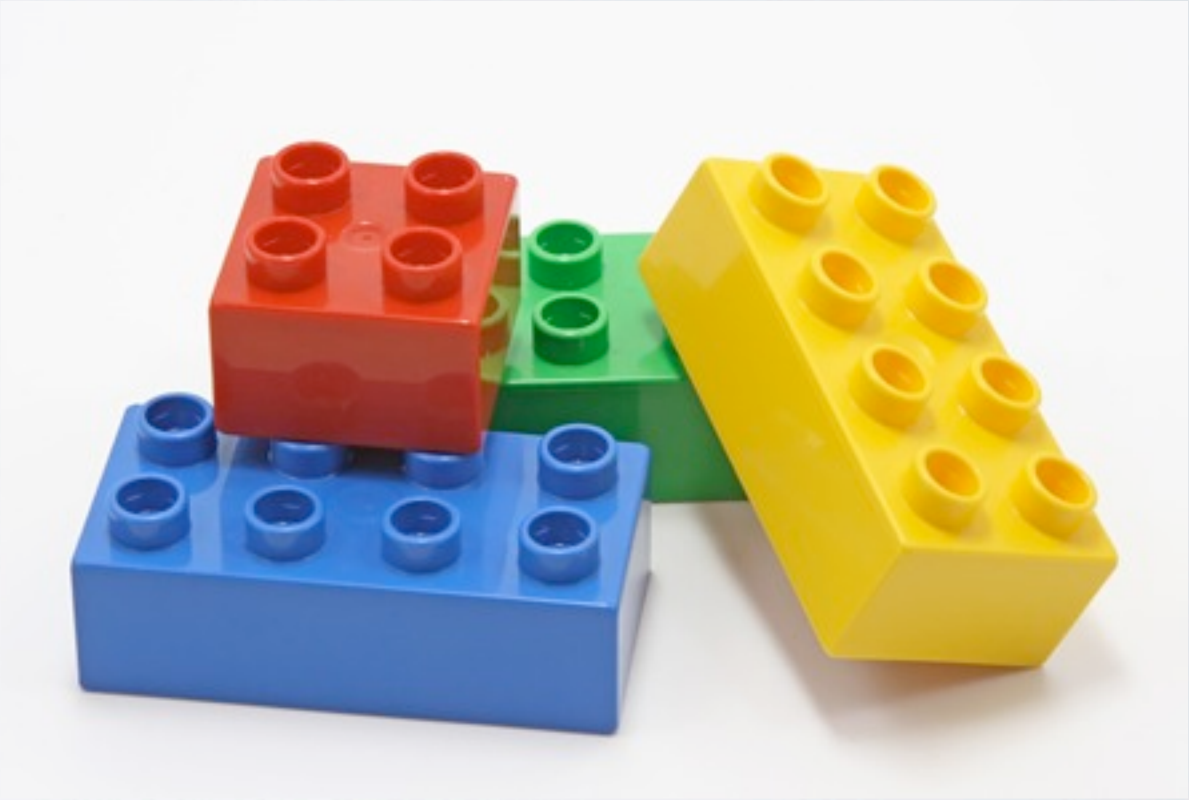
The focus is on refining and studying the business-based aspects of the computer system.

PHASES OF DSDM

Design and Build Iteration

The product is designed and developed in iterations.

In each iteration a design model is made of the area being developed, and then that area is coded and reviewed.



PHASES OF DSDM

Implementation

Product is wrapped up.

Documentation is written.

Review is drawn up.

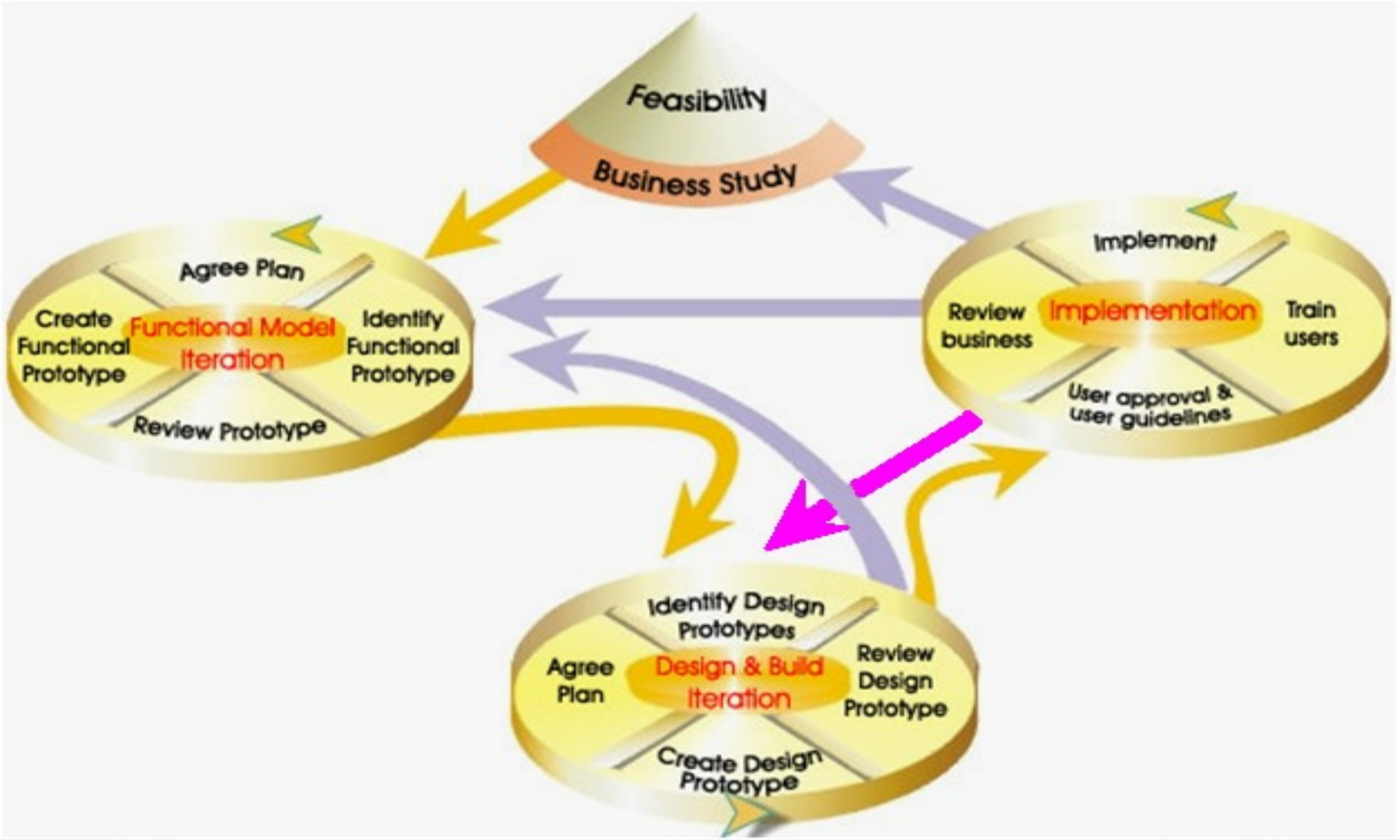
Compare the requirements with their fulfillments in the product.

The users are trained in how to use the system, and the users give approval to the system

PHASES OF DSDM

Post-project

Post-project tasks include measurements on how the deployed system is performing and if any further enhancements are required.



PROJECT LIFE CYCLE OVERVIEW

ADVANTAGES OF DSDM

- Active user participation throughout the life of the project and iterative nature of development improves quality of the product
- Ensures rapid, effective and maintainable deliveries which match the needs of the business better
- Both of the above factors result in reduced project costs

LIMITATIONS OF DSDM

- Requires significant user involvement
- Switching to DSDM is neither cheap nor fast, and requires a significant cultural shift in any organization

COMPANIES USING DSDM

BRITISH AIRWAYS

SCRUM

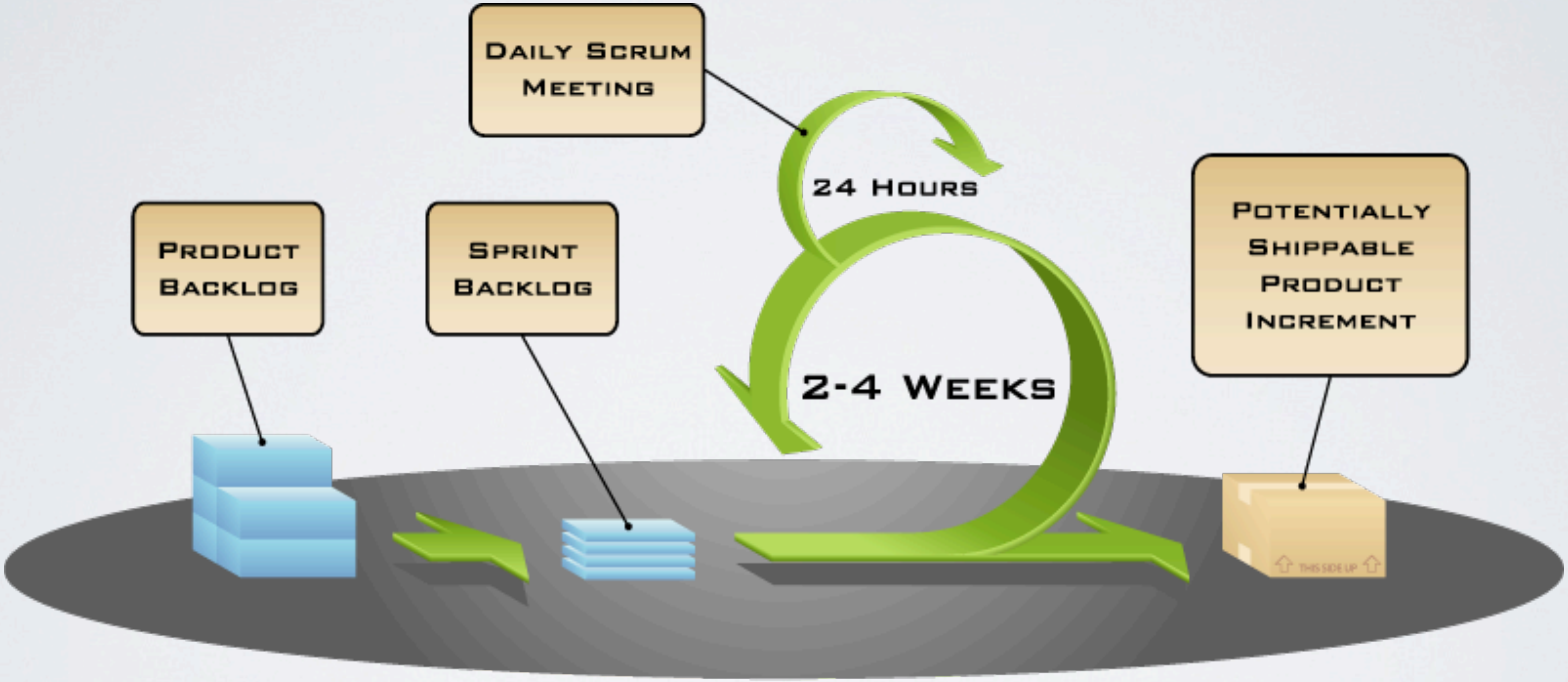
HISTORY

- Started out as “rugby approach”
  - Applied to commercial product development
  - Team tries to “go the distance as a unit, passing the ball back and forth”
  - Emphasized speed & flexibility
- Became “Scrum,” referring to a restart in rugby, or the huddle

WHAT IS SCRUM?

- Not just a standup meeting every day
- "An agile framework for completing complex projects." – Scrum Alliance
- Iterative and incremental process
- Repeatable way to divide larger projects into manageable pieces
- Projects are divided into pieces called Sprints
- Teams are self-organizing and cross-functional
    - No reliance on other groups to deliver
    - Team members under different disciplines are physically close to each other
- 3 main lists (artifacts), meetings, and roles

OVERVIEW OF THE PROCESS

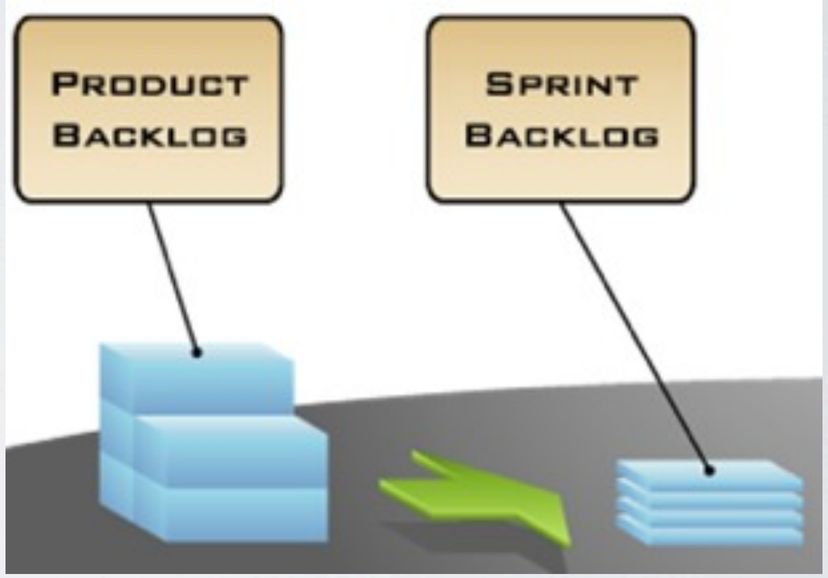


SPRINTS

- Typically last 1 to 4 weeks
- Preceded by a Sprint planning meeting
- Team works on completing the items in the Sprint backlog
- No changes allowed to the Sprint backlog once the Sprint has started
- At the end of every Sprint, the team presents a usable product
- Sprint review meeting
- After the current Sprint is finished, begin the next one

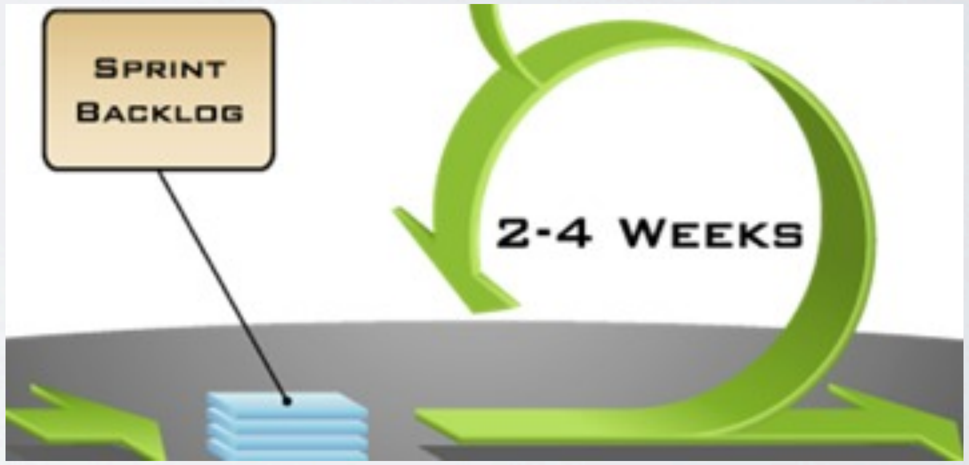
ARTIFACTS: PRODUCT BACKLOG

- Prioritized list of everything needed in the product
- Single source of requirements for any changes to be made to the product
- Constantly evolving and changing
- Attributes - description, order, estimate
- May contain features, bug fixes, requirements, etc.
- As product is used and feedback acquired, product backlog will grow

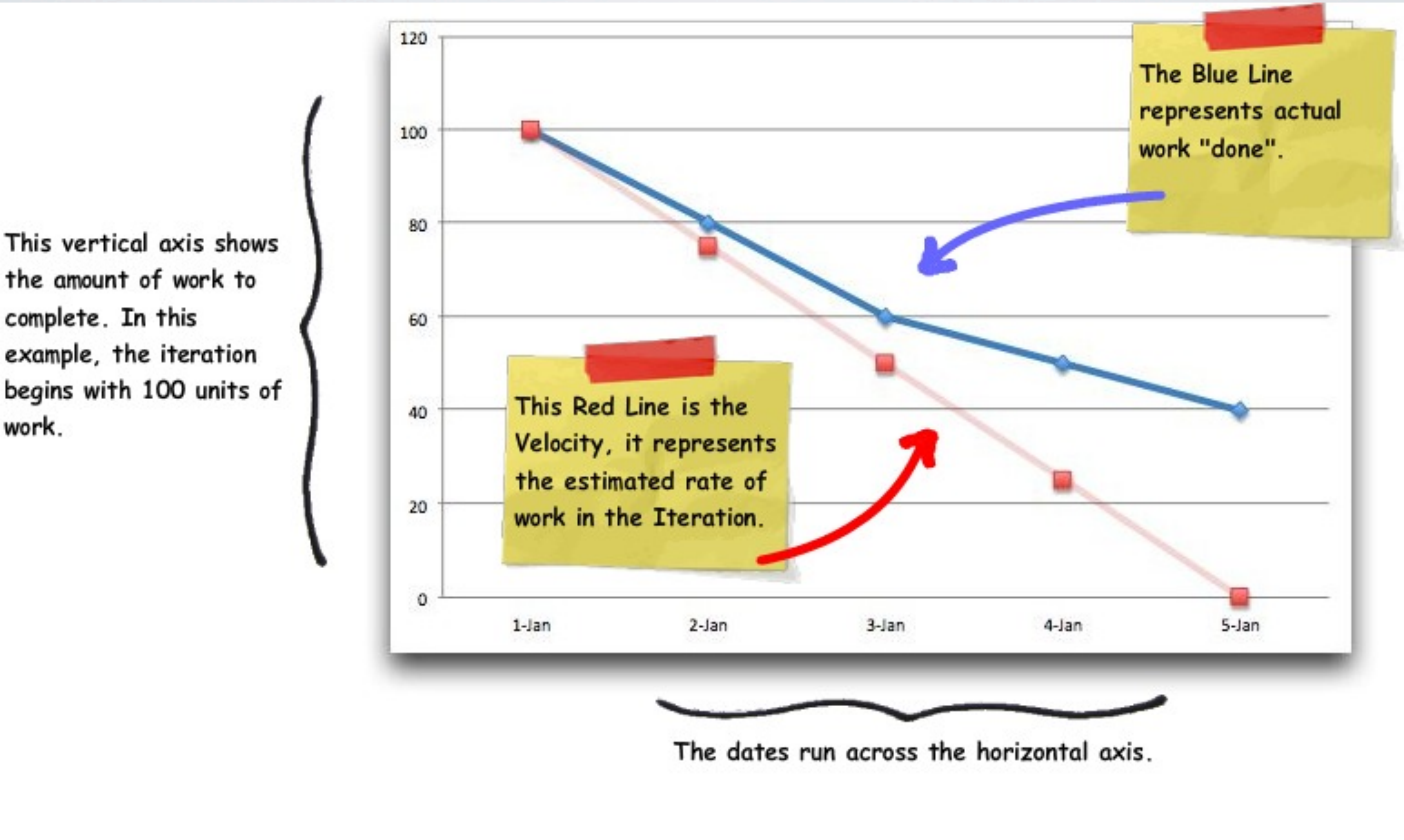


ARTIFACTS: SPRINT BACKLOG

- The set of Product Backlog items selected for the current Sprint
- A forecast of what functionality will be in the next increment of work
- Modified throughout the Sprint
- Belongs to the Development Team only
  - Real-time picture of the team's status
  - Only the team can add/remove items



ARTIFACTS: BURNDOWN CHART



This vertical axis shows the amount of work to complete. In this example, the iteration begins with 100 units of work.

The Blue Line represents actual work "done".

This Red Line is the Velocity, it represents the estimated rate of work in the Iteration.

The dates run across the horizontal axis.

ROLES: PRODUCT OWNER

- Typically a single person
- Represents the voice of the customer – communicates the vision of the product to the Development Team
- Maximizes value of the product and work of the Development Team
- Responsible for delivering the best possible product within time and budget
- Solely responsible for managing the Product Backlog
  - Content
  - Organization
- Organization must respect the Product Owner's decisions (visible in the Product Backlog's content/organization)
- Similar to a sports coach – sets up the plan, focuses the team on the goal and then lets them execute it

ROLES: DEVELOPMENT TEAM

- Responsible for delivering a potentially shippable product at the end of every Sprint
- Cross-functional skills ensure independence from other groups
- Self-organizing (still may meet with project management) – only ones who decide how to turn backlog into releasable increments
- No titles (other than Developer)
- No sub-teams
- Optimum size is 3-9 members
  - Smaller: skill constraints, smaller increments
  - Larger: excessive coordination and complexity

ROLES: SCRUM MASTER

- Servant-leader for the team
- Makes certain the Scrum process is understood
- Ensures the team adheres to Scrum theory and practices
- Enforces the rules - part of the Scrum Master’s job is as the referee
- Remove obstructions in the team’s way
- Buffer between the team and distracting influences
- Acts like a liaison between Product Owner and Development Team

MEETINGS: SPRINT PLANNING

- Designates what work will be done in the current Sprint
- Takes items from the top of the Product Backlog and places them in the Sprint Backlog
- Product Owner presents the Product Backlog items to the Development Team
- Development Team solely decides the number of items to be placed into the Sprint Backlog
- Product Owner can clarify selected Product Backlog items and help make trade-offs
- Development Team can renegotiate work with Product Owner if there is too much/little

MEETINGS: DAILY SCRUM

- Usually a stand-up meeting that lasts no longer than 15 minutes
- Assesses daily progress made by the Development Team
- Only Development Team members speak (others may attend)
- Each member of Development Team:
  - What has been accomplished since last meeting?
  - What will be done before next meeting?
  - What obstacles are in the way?

- Scrum Master's role is to facilitate the resolution of these obstacles/impediments and keep the meeting under 15 minutes

MEETINGS: SPRINT REVIEW

- Performed after Sprint is finished

- Review work completed/incomplete
- Present/demonstrate completed product to stakeholders
- Incomplete work is not demonstrated
- Discuss what went well, which problems arose, and how they were solved
- Product Owner discusses Product Backlog as it stands
- Entire group collaborates on what to do next and prepare for the next Sprint Planning meeting
- End result: revised Product Backlog

OTHER MEETINGS

- Sprint Retrospective
- Scrum of Scrums
- Backlog grooming

SPRINT RETROSPECTIVE

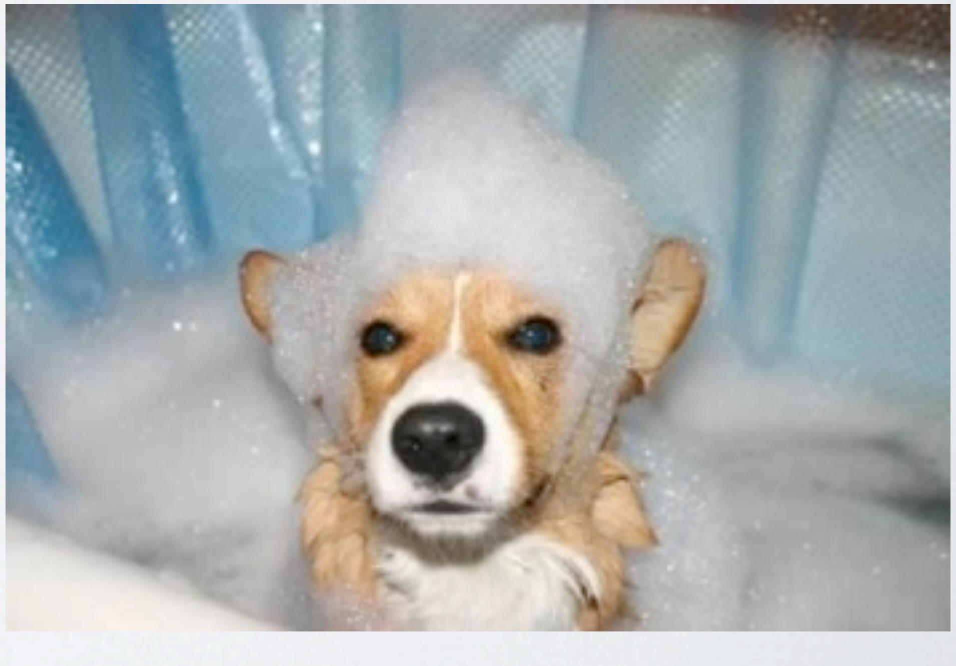
- Team members reflect on the past sprint
- Lessons learned
- Process improvements – plan to improve the next Sprint

SCRUM OF SCRUMS

- Used in scaling Scrum to large project teams
- A representative from each Daily Scrum Meeting conducts their own Scrum Meeting
- What has your team done since we last met?
- What will your team do before we meet again?
- Is anything slowing your team down or getting in their way?
- Are you about to put something in another team's way?

BACKLOG GROOMING

- Team spends time during the Sprint to do backlog grooming
- Estimating backlog effort
- Refining acceptance criteria for backlog items



PROS

- Breaks down projects into smaller, digestible chunks in a methodical way
- Each team member carries responsibility and ownership
- Increased communication
  - Daily Scrum Meetings
  - Physical location of team members
- Provides quantified progress
- Constant and current status updates (can be a con)
- Customer can update/change requirements
- Constant feedback from customer through regular demonstrations of iterations
- Low up front documentation
- Developers can choose what they take on

CONS

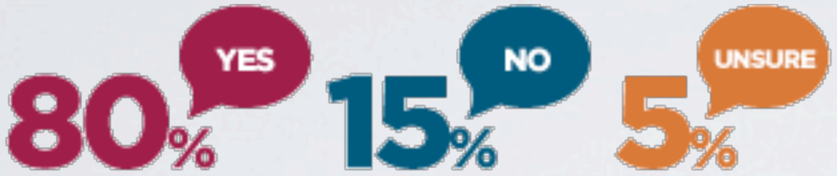
- Even with Scrum of Scrums Meetings, large scale projects may present complications. For example, system testing may become difficult.
- Meetings and planning can be excessive – takes time away from development
- Have to update status constantly
- Shorter iterations have higher overhead
- Possibility of scope creep (items keep getting added to Project Backlog)

WRAP UP

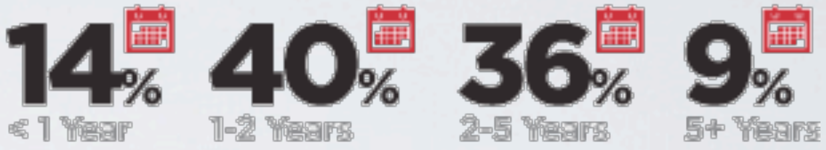
- Adoption of Agile in the Industry

Source: VersionOne - "The State of Agile Development" 2011

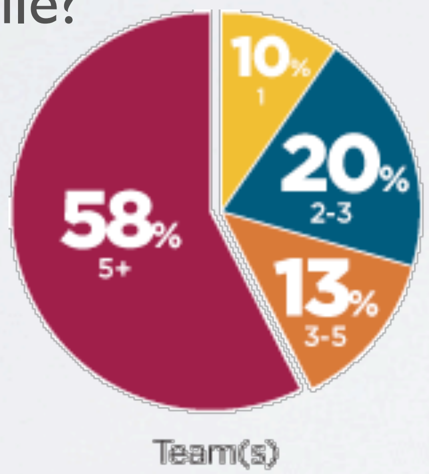
How Many Organizations have adopted agile development?



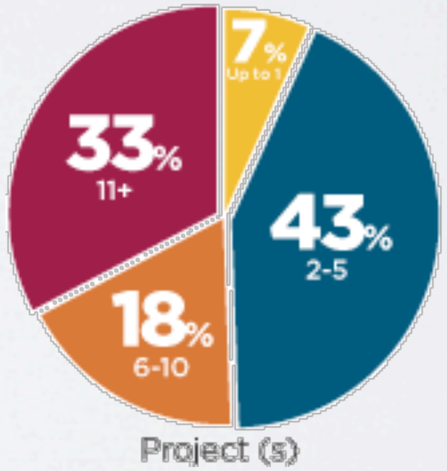
How long have respondents been practicing agile development?



How Many Teams Adopted Agile?



Number of Company Projects Using Agile



WHO'S USING AGILE?

Financial Services

M

HOOVERS
A D&B COMPANY

STANDARD & POOR'S
RATINGS SERVICES

Healthcare

McKESSON

Carestream

Harvard Pilgrim Health Care

WOLF
MEDICAL SYSTEMS

PINNACOL
ASSURANCE

Government

DEPARTMENT OF THE NAVY
UNITED STATES MARINE CORPS

LOCKHEED MARTIN

NORTHROP GRUMMAN

vsti

Raytheon

United States™
Census
2010

Media & Entertainment

The Walt Disney Company

gazillion
entertainment

NBCUniversal

The McGraw·Hill Companies

Sony Ericsson

Regulated Industries

DEPARTMENT OF DEFENSE
UNITED STATES OF AMERICA

perotsystems

Tech

bmcsoftware

Microsoft

YAHOO!

ORACLE

intel

amazon.com

ebY

salesforce
SOFTWARE

hp

Telecom

Level(3)
COMMUNICATIONS

Comcast

AVAYA

WHO'S USING AGILE?

Q&A

Keith Pine • Kumeel Alsmail • Parker Li • Björn Davis

REFERENCES

- Ambysoft - http://www.ambysoft.com
- State of Agile Development Survey Results - http://www.versionone.com/state_of_agile_development_survey/11/
- Manifesto for Agile Software Development - http://agilemanifesto.org/
- Marting Fowler, The New Methodology - http://martinfowler.com/articles/newMethodology.html

REFERENCES

- Williams, Laurie, A Survey of Agile Development Methodologies - http://agile.csc.ncsu.edu/SEMaterials/AgileMethods.pdf
- Rally Software - http://www.rallydev.com
- Mountain Goat Software - Reported Benefits of Agile Development
- Wysocki, Robert K., 2009. *Effective Project Management.*
- Slideshare.net