Software Processes

Lecture 2 – “Just do” or “Do It!”

Jesper Andersson

Software development: What are the Goals?

Developer

Vendor

Profit

Services

Maximize!

WinWin!

Training

Quality

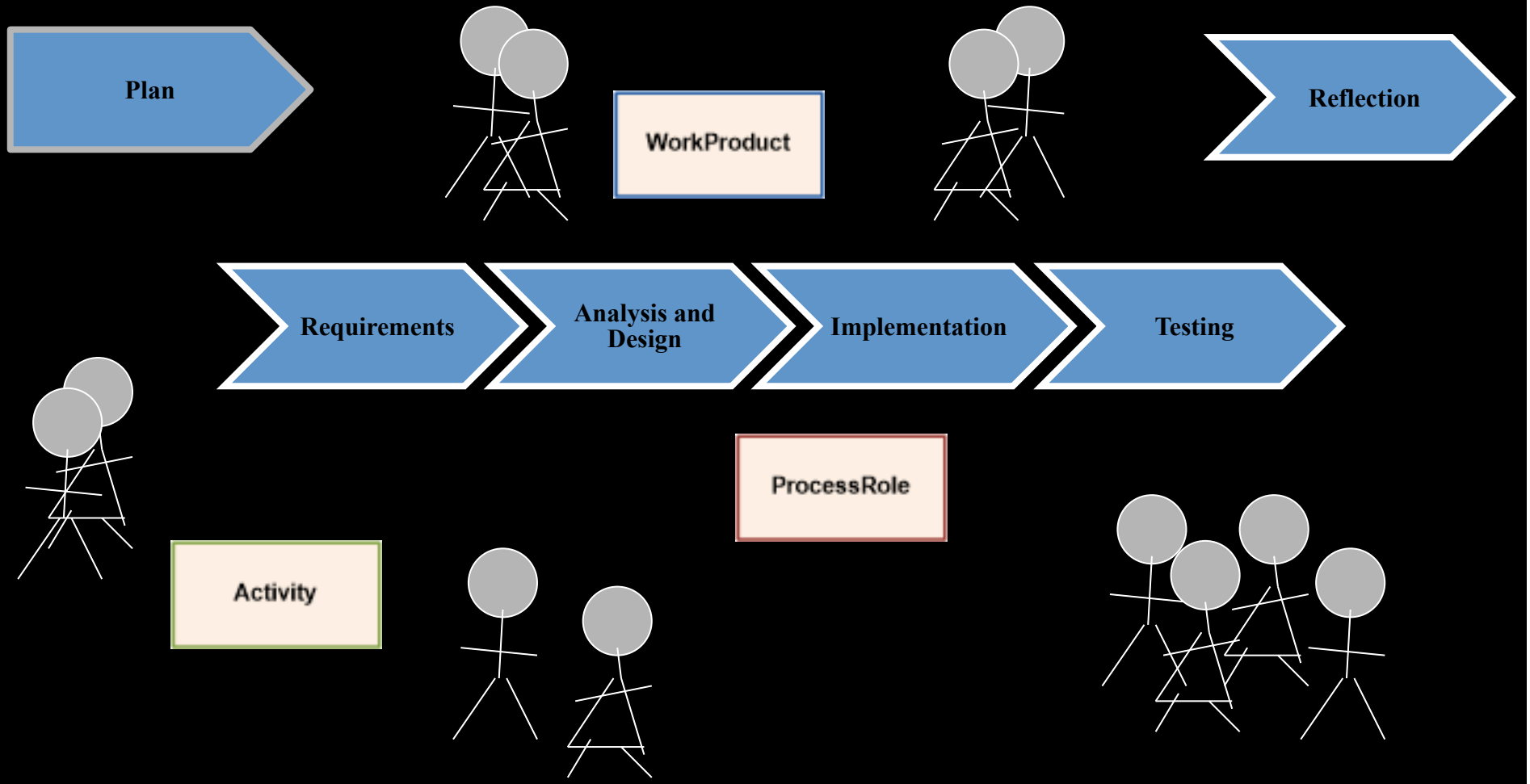
Cost

Time

End user

Procurer

Principle Challenge: Gain Control and Keep it!



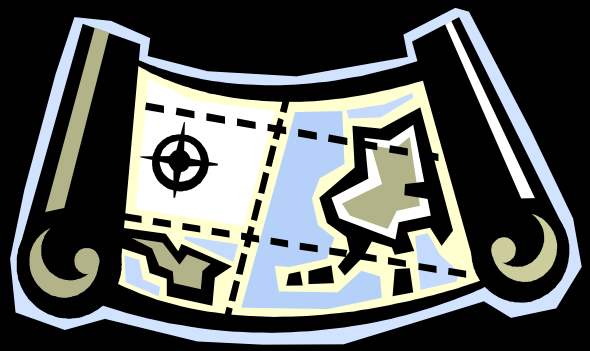
The Need for a Process

- ✓ In industry all forces must work **in the same direction**
- ✓ The work must be **Controllable**
- ✓ The work must be **Repeatable**
- ✓ The results should be similar in character

- ✓ Gain control and Keep it!

Software Processes

- ✓ Roadmap for successful software development
  - What
  - When
  - Who
  - How
  - Why



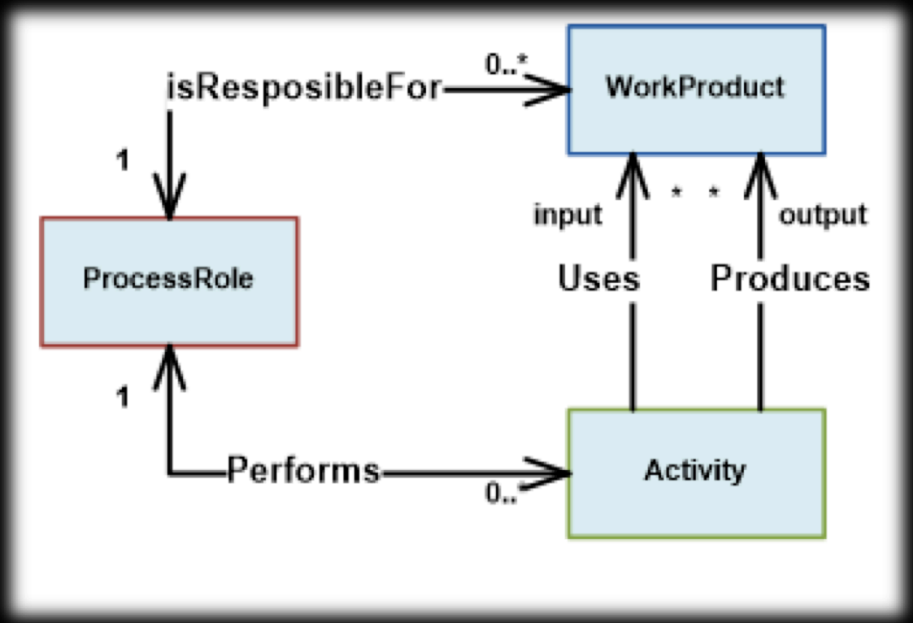
- ✓ We gained control and may keep it!

How do we Organize Work?

- ✓ *Roles* – Resources, Skills and Time
- ✓ *Activities*
- ✓ *Work Products* – Artifacts

- ✓ Factors
  - Time
  - Dependencies
  - Competences
  - Resources

➔ **Process model**



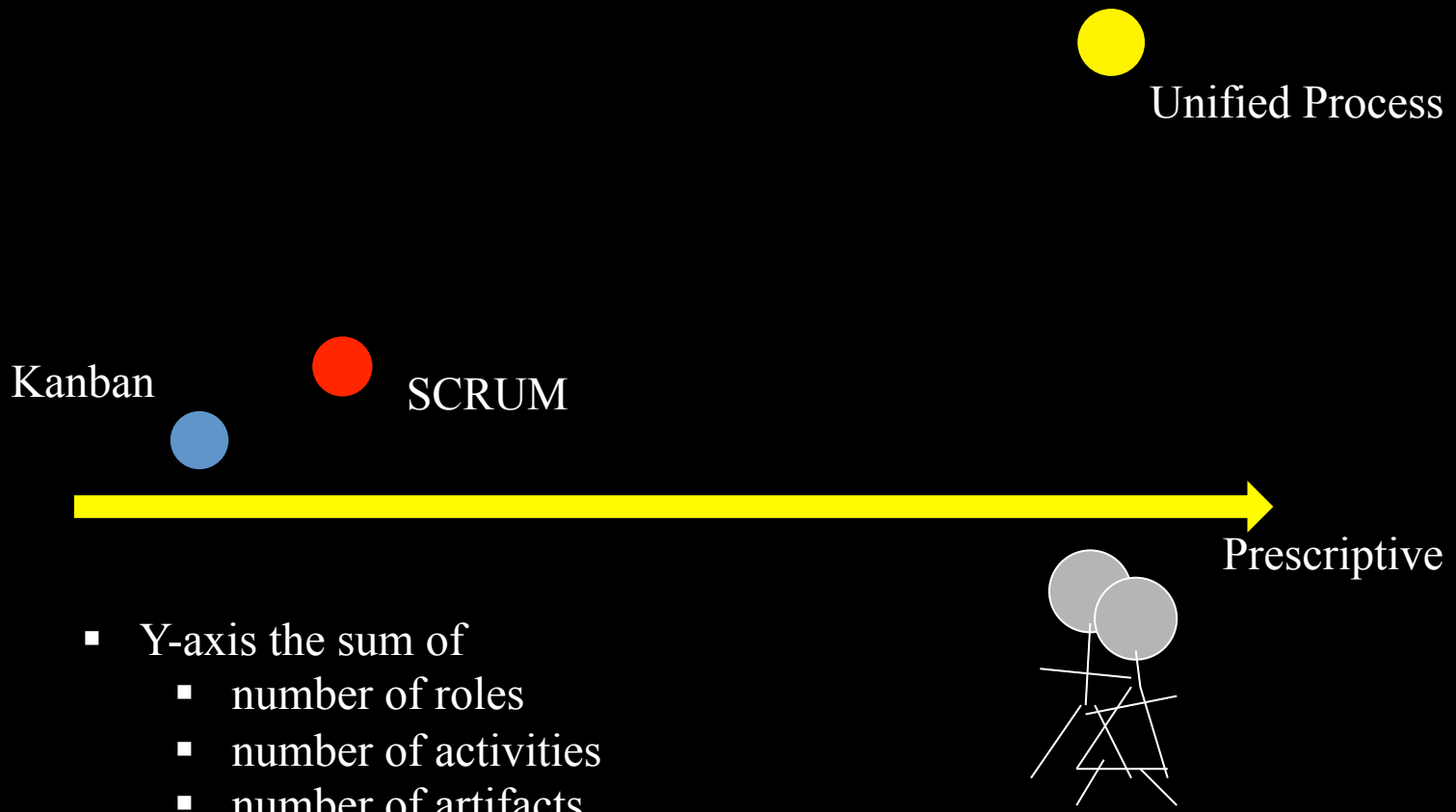
Software Development Activities

- ✓ Requirements
- ✓ Analysis
- ✓ Design
- ✓ Implementation

- ✓ Testing
- ✓ Integration
- ✓ Evolution

They are always there!

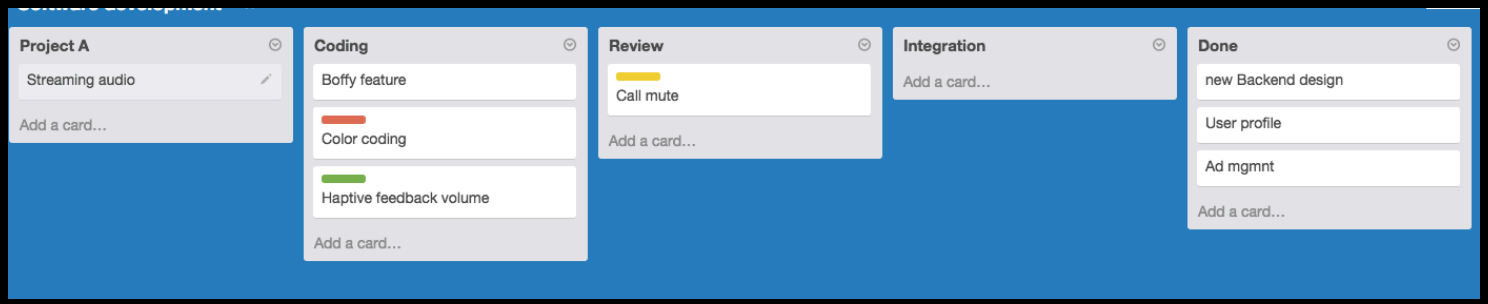
Process Models – Many Flavors



- Y-axis the sum of
  - number of roles
  - number of activities
  - number of artifacts

Example: Process elements in Kanban

- ✓ Background – Lean production
- ✓ No “iterations”
- ✓ No “roles”
- ✓ Simple Workflows
- ✓ Visual



Example: Process elements in UP

Role Name	Project Deliverable List	Software Development Plan	GANTT Chart	Status Report	Issue Log	Risk List	Glossary	Use Cases	Candidate Architecture	Object Model	Physical Data Model	Design Guidelines	RUP Worker Matrix	Test Cases	Test Plan	User Interface Design Specifications	Individual Name
Analyst Worker Set																	
Business-Process Analyst																	
Business Designer																	
Business-Model Reviewer																	
Requirements Reviewer								C						C		C	
System Analyst		C						C			C						
Use-Case Specifier								R									
User-Interface Designer																R	
Developer Worker Set																	
Architect		C				C		C			C	R		R		C	
Architect Reviewer		C										C					
Capsule Designer																	
Code Reviewer																	
Database Designer		C						C			R	C					
Design Reviewer																	
Designer		C									C	C					
Implementer		C						C			C	C					
Integrator		C									C	C					
Tester Worker Set																	
Test Designer																	
Tester		C						C						C			
Manager Worker Set																	
Change Control Manager																	
Configuration Manager																	
Deployment Manager																	
Process Engineer																	
Project Manager		R	R	R	R	R									C		
Project Reviewer		C				C		C									
Additional Worker Set																	
Any Worker																	
Course Developer																	
Graphic Artist																	
Stakeholder																	
System Administrator																	
Technical Writer																	
Tool Specialist																	

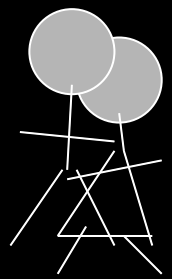
Example: Process elements in SCRUM

- ✓ Roles
  - Product Owner – owner of the product vision
  - Scrum Master – assist the team in their application of SCRUM
  - Development team – builds the product
- ✓ Artifacts
  - Product increment – a finished part of a product.
  - Product backlog – a prioritized list of ideas for a product
  - Sprint backlog – a detailed plan for the development during the next sprint.
- ✓ Ceremonies
  - Sprint Planning – planning in the beginning of a sprint
  - Sprint Review – ”demonstration” of sprint result
  - Sprint Retrospective – ”sprint” post-mortem, improvement meeting
  - Daily Scrum Meeting – short daily meeting with team and scrum master

Again: What's lurking under the surface?

- ✓ Requirements
- ✓ Analysis
- ✓ Design
- ✓ Implementation

- ✓ Testing
- ✓ Integration
- ✓ Evolution

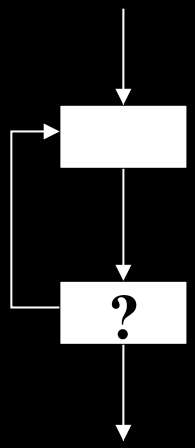


Some principles to follow when organizing work

- ✓ Mile-stones
- ✓ Evolutionary
- ✓ Iterative and incremental
- ✓ Value-driven

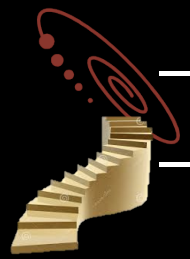
Evolutionary – when you are lost or uncertain

- ✓ Exploratory
  - System evolves as developers understand the problem domain
- ✓ Throw-away prototyping
  - A prototype is used for requirements verification. The prototype is evaluated and used as input to the requirements specification activity.



Let us bring Two Principles together

- ✓ Challenges
  - Complex systems
  - Complex teams
  - Resource shortage
  - Additional constraints
- ✓ Solutions
  - Cycles
  - Shorter-time frames
  - Piece-wise addition
  - Reduced problem sizes
  - **Better control!**
  - Iterative and
  - Incremental



With Two Principles together, we gain control

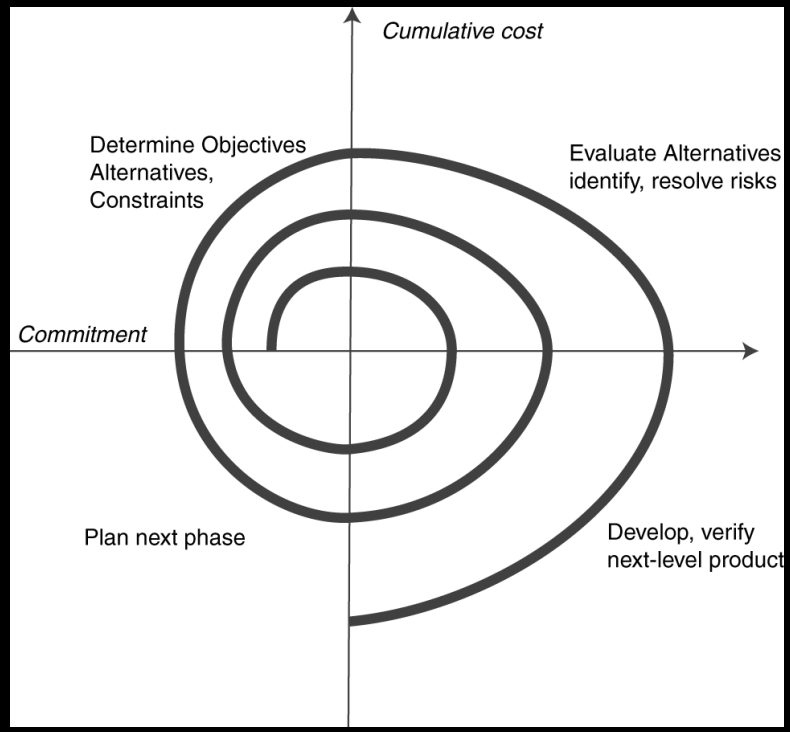
- ✓ Iterative
  - Repeat activities
  - Mini-projects
  - Shorter time-frames
- ✓ Incremental
  - Piece-wise addition
  - Reduced problem sizes

Drivers

- ✓ What are deciding your targets?
  - Value (for company)
  - Time
  - Value (for customer)
  - Risk
  - Progress (artifact sign-off)

- ✓ These “drivers” are included in the decisions about
  - Iterations, “what should we do and when”
  - Increments, “what are the (current) focus of our work”

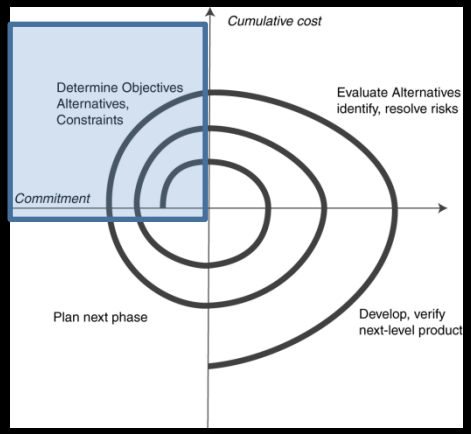
Iterative models – Spiral Model



- ✓ A generic process model, i.e. not only for software projects
- ✓ Risk driven (engineering principles)

Spiral Model - First Quadrant

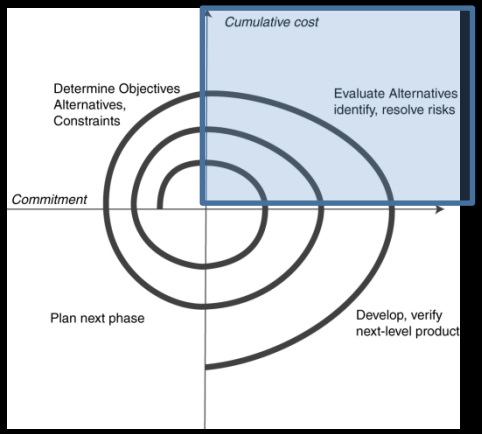
- ✓ Specify the goals for this part of the system or project
- ✓ Identify alternative paths to reach the specified goals
- ✓ State restrictions put on this part of project or process



1

Spiral Model - Second Quadrant

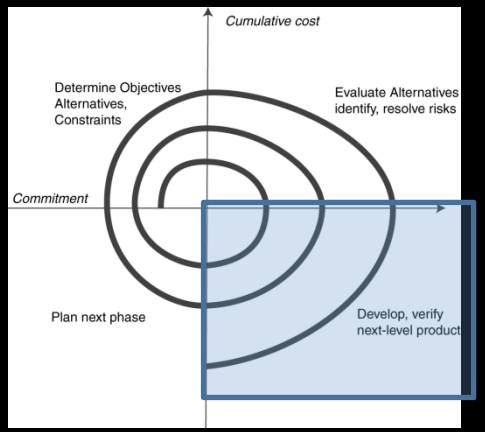
- ✓ Identify risks associated with this part of the project or the process
- ✓ Evaluate the alternatives to minimize risk
- ✓ Plan for how to manage remaining risks



2

Spiral Model - Third Quadrant

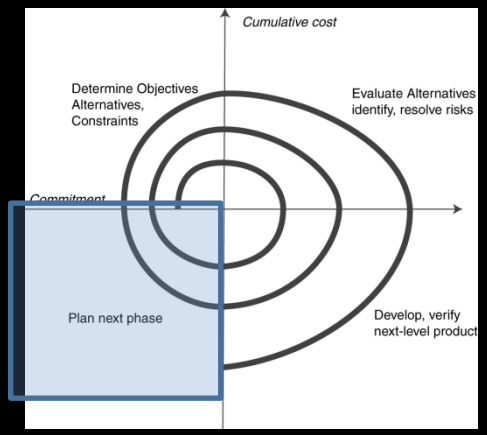
- ✓ Carry out **one** or **some** of the alternatives
- ✓ Evaluate!



3

Spiral Model - Fourth Quadrant

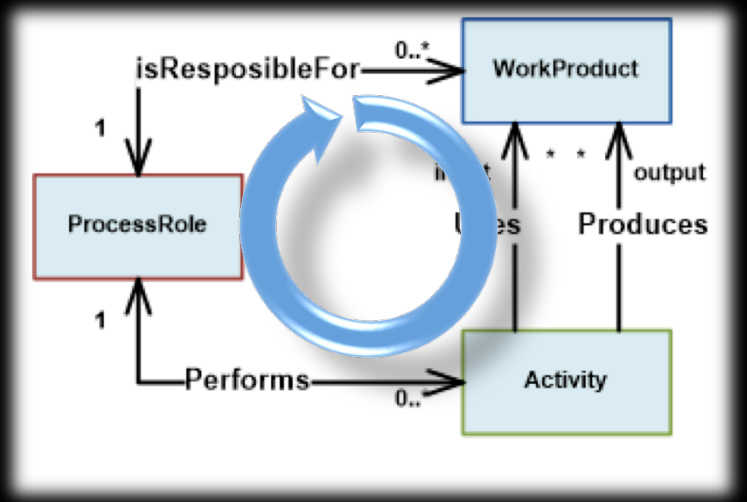
- ✓ Based on the evaluation in the previous quadrant, plan for the next iteration
- ✓ In this quadrant the project group can decide to exit the spiral (finished project)



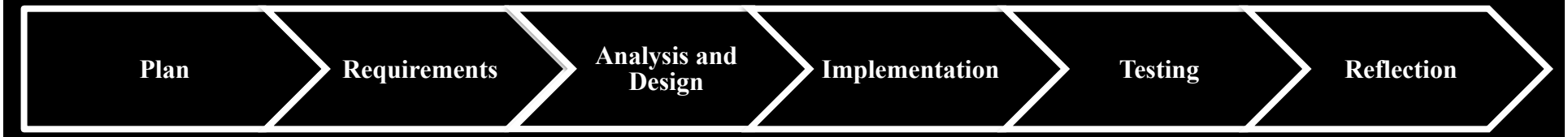
4

Iterations

- ✓ Mini-project
  - Planning
  - Analysis & Design
  - Construction
  - Internal or External Release
- ✓ Generates an increment



Example – File transfer



Problem: Develop an application that transfers files from A to B

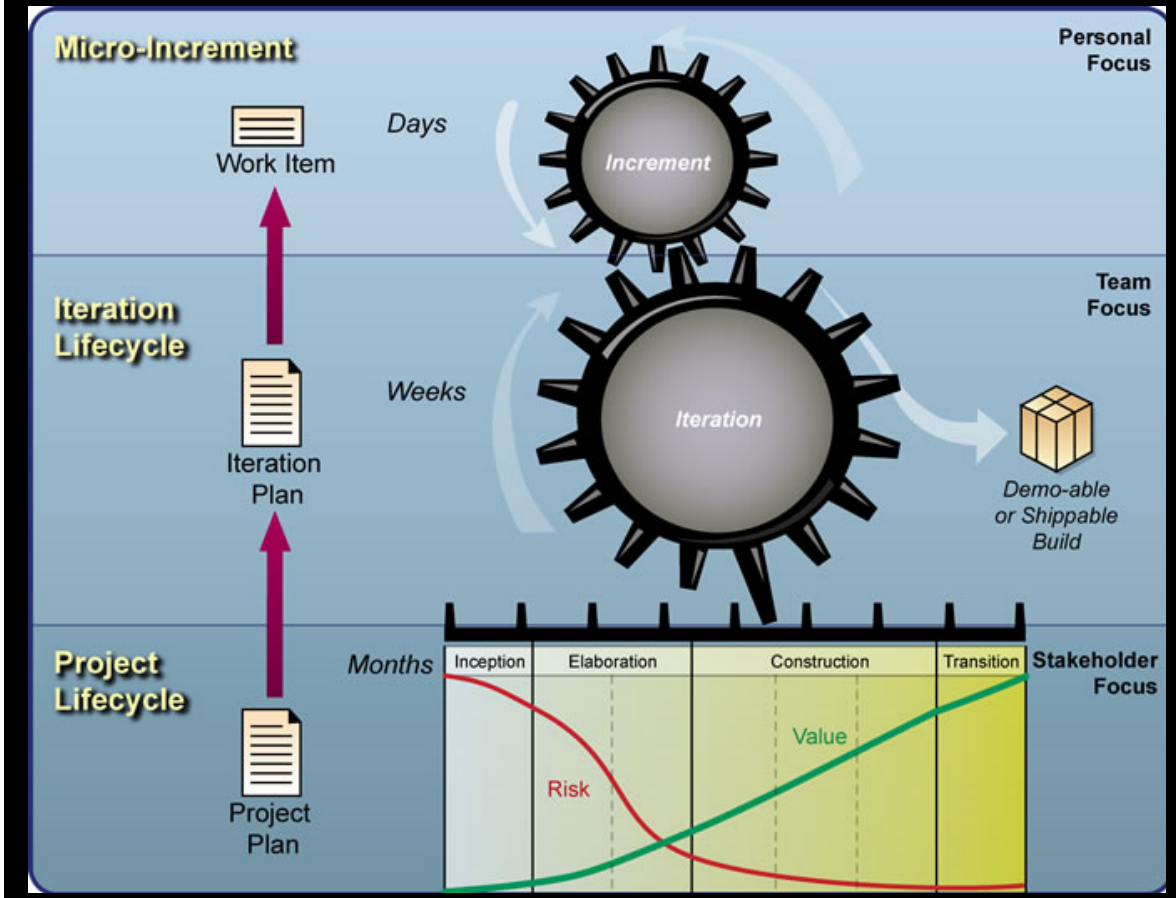
Increment #1 – Data transfer
Increment #2 – File management

Increment #1 + Increment #2 = Solution

Iteration #1 – Establish Data connection
Iteration #2 – Data transfer
Iteration #3 – File management
Iteration #4 – File transfer

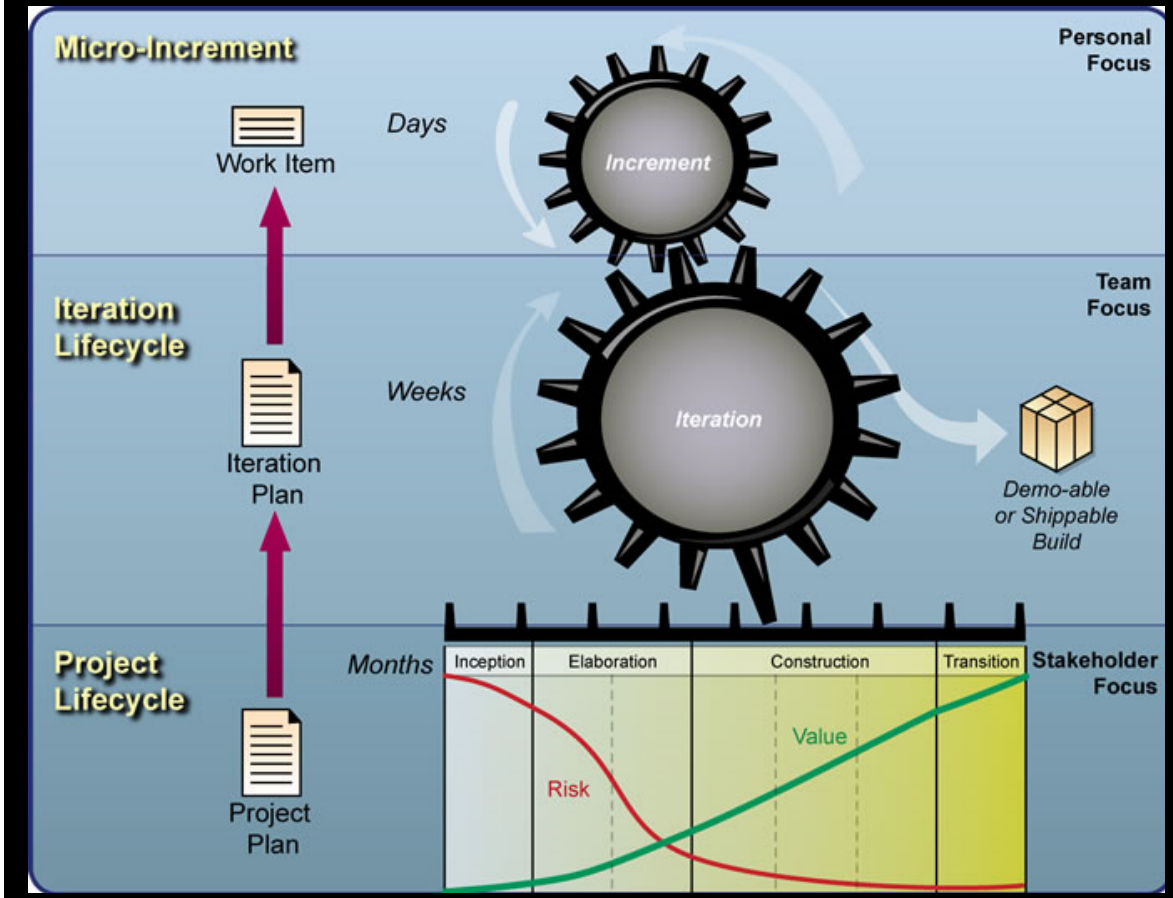
Each iteration produces something that is executable
However, not necessarily feature complete (incremental)

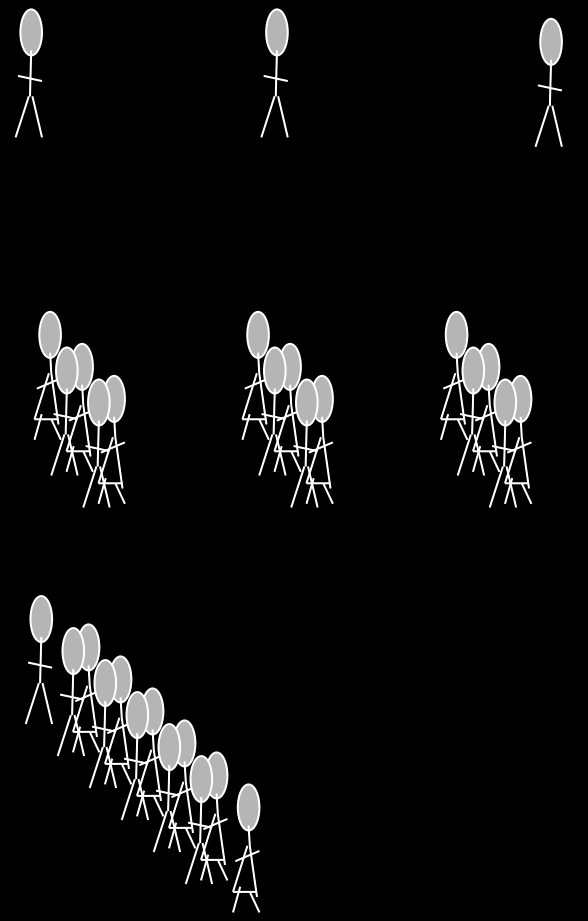
Iterations and Increments – Open UP



Source: http://epf.eclipse.org/wikis/openup/

Iterations and Increments – On each level!

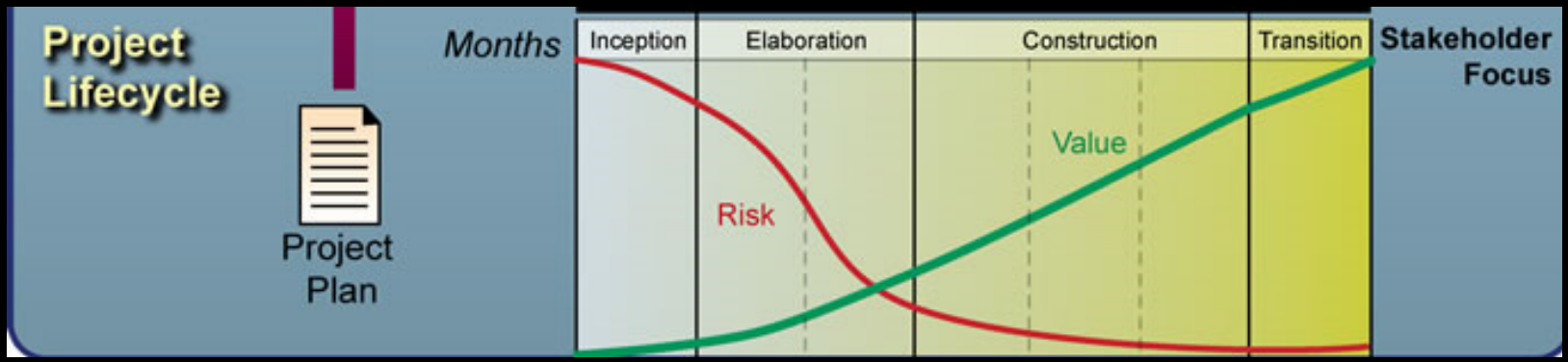




Source: http://epf.eclipse.org/wikis/openup/

OpenUP - Phases

- ✓ High-level increments with generic goals
  - Inception – Get the project off the ground
  - Elaboration – Executable architecture baseline
  - Construction – Feature complete system
  - Transition – Perfected system



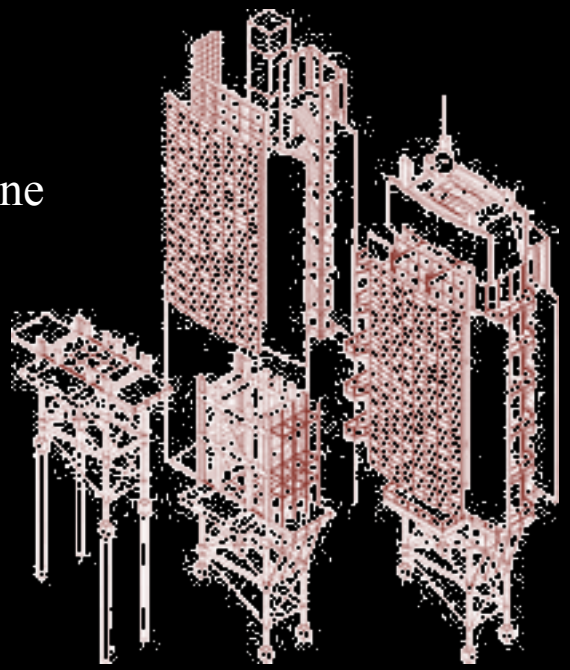
Source: http://epf.eclipse.org/wikis/openup/

Inception

- ✓ Goals
  - Launch the project
  - Develop an evolvable set of artifacts
- ✓ Focus
  - Customer
  - Project
  - Risk assessment
- ✓ Mile-stones
  - Define significant use cases
  - Define candidate architecture
  - Setup project environment

Elaboration

- ✓ Goals
  - Establish an executable architecture baseline
  - Detailed construction plan
- ✓ Focus
  - Risk reduction
  - Requirements analysis
  - Initial architecture implementation
- ✓ Mile-stones
  - Executable architecture baseline



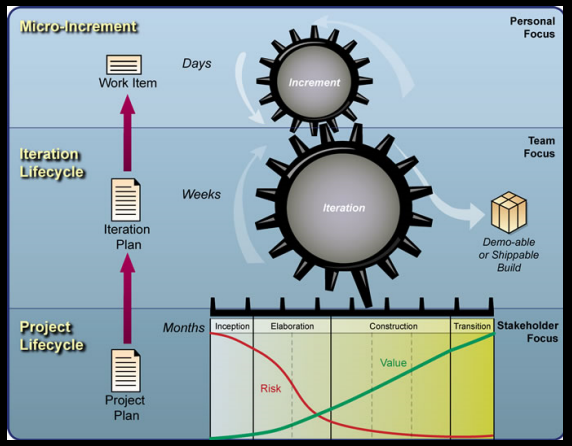
Construction

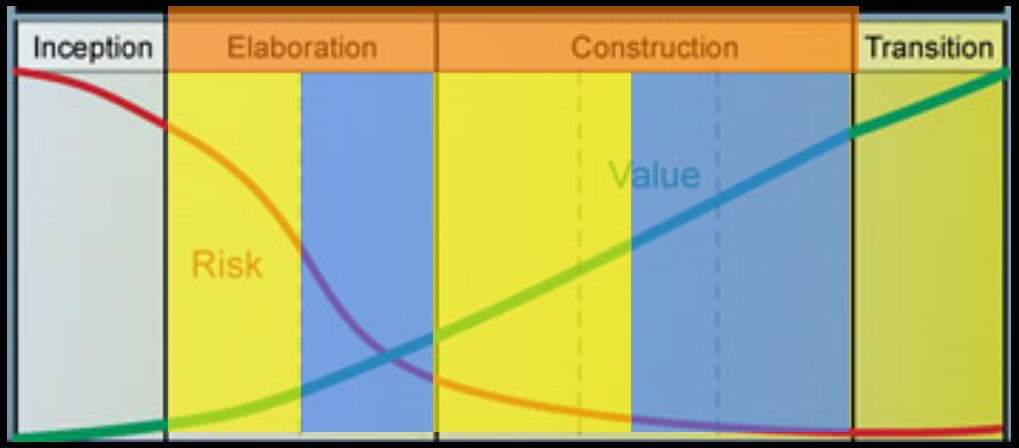
- ✓ Goals
  - Evolve architecture baseline into final system
- ✓ Focus
  - Value Creation
  - Implementing features
  - Test features
- ✓ Mile-stones
  - Feature complete
  - Frozen software system

Transition

- ✓ Goals
  - Perfecting the system
  - Find and Correct defects
  - Perform user specific adaptations
- ✓ Focus
  - Maintain design & implementation
  - Beta tests & Acceptance
  - SYSTEM RELEASE!

Iterations and Increments – Open UP Elaboration and Construction Phases



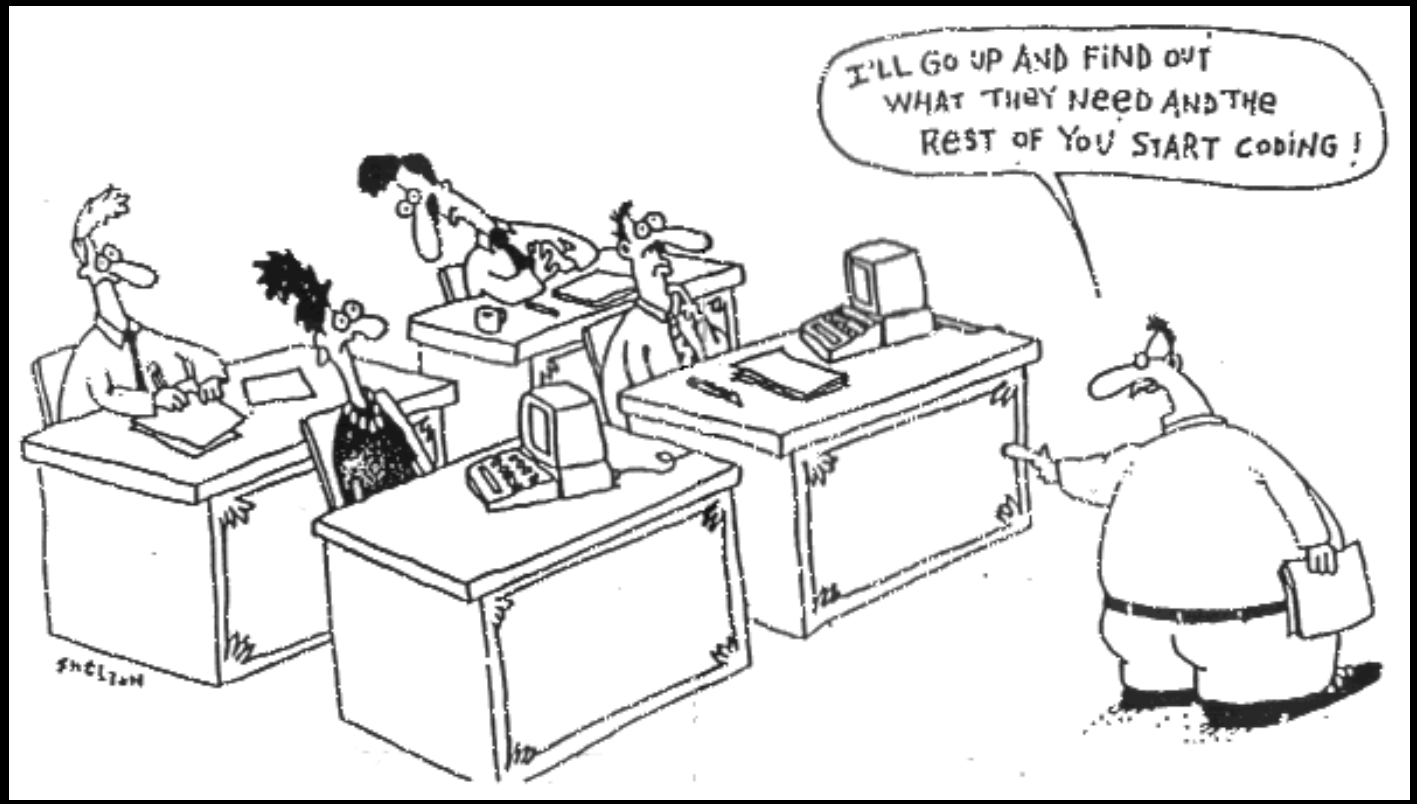


Increment #1 – Data transfer
Increment #2 – File management

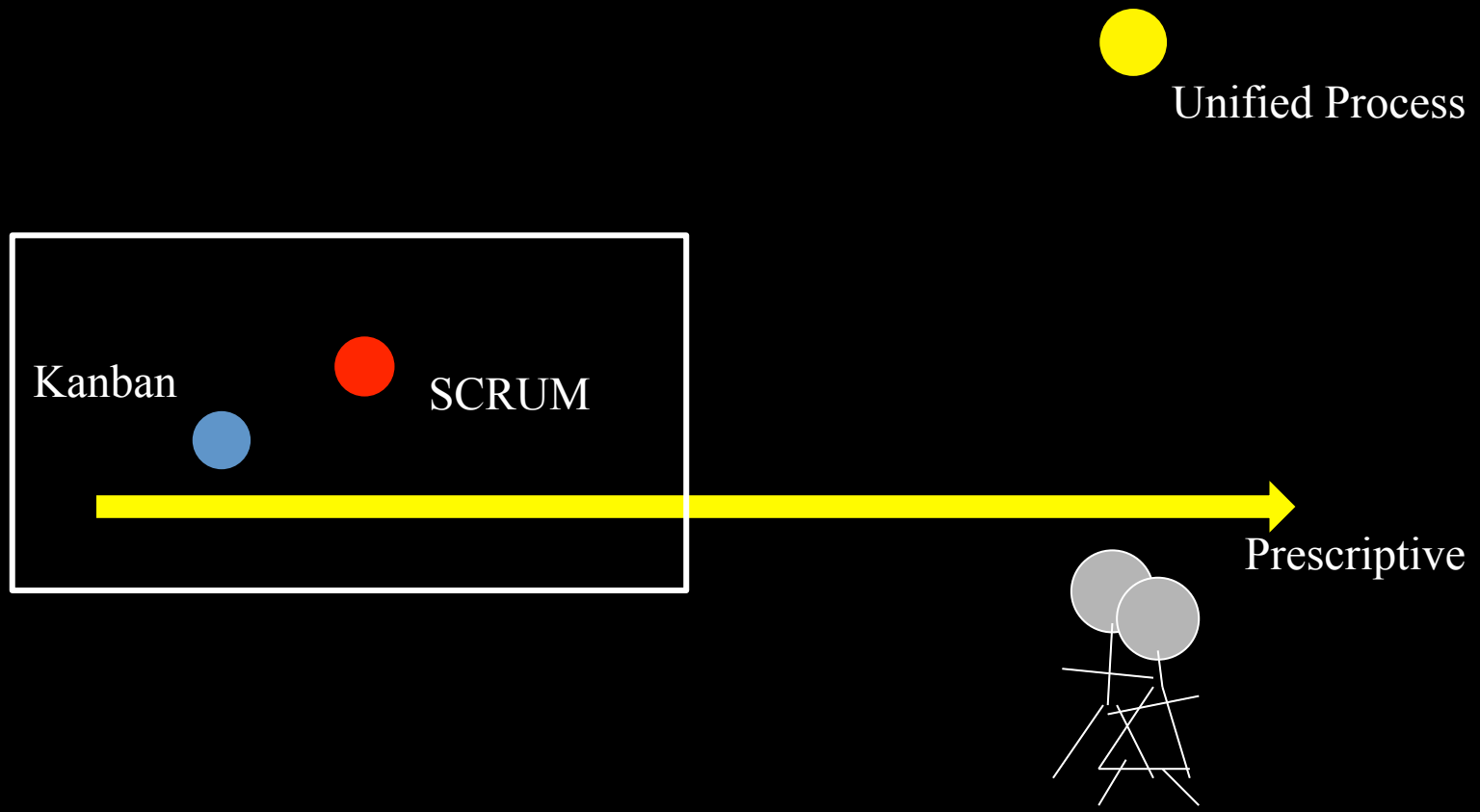
Increment #1 + Increment #2 = Solution

Iteration #1 – Establish Data connection
Iteration #2 – Data transfer
Iteration #3 – File management
Iteration #4 – File transfer

Plan-driven or Agile? – Capture the flag!



Process Models – Many Flavors



Target – Value Creation

- ✓ Minimum Viable Product

- ✓ A product composed of olyn the most necessary features.
- ✓ What creates the highest return on investment at the lowest risk.

- ✓ Lean development – minimize waste
  - Define: *Waste*
  - *Unnecessary work*
  - No value for customer

Agile manifesto

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

- ✓ That is, while there is value in the items on the right, we value the items on the left more.

source: http://agilemanifesto.org/

Agile Manifest principles

- ✓ 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.

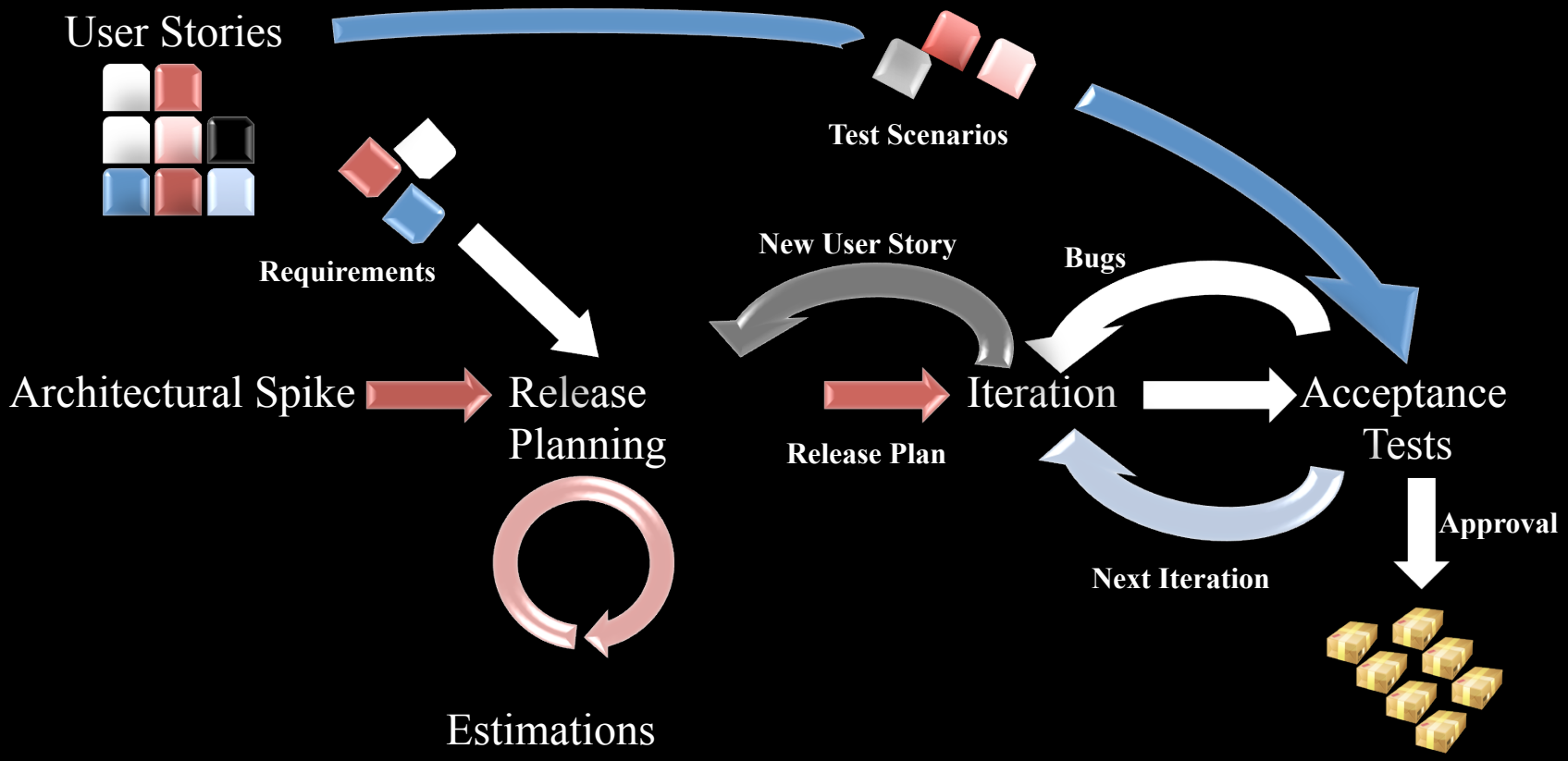
source: http://agilemanifesto.org/

Agile Manifest principles

- ✓ 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.

source: http://agilemanifesto.org/

XP


User Stories
Test Scenarios
Requirements
New User Story
Bugs
Architectural Spike
Release Planning
Release Plan
Iteration
Acceptance Tests
Approval
Next Iteration
Estimations
Small Releases

eXtreme Programming Principles & Practices

- ✓ eXtreme Programming describes in total 12 agile “methods” or engineering practices
  - 40 hour work week
  - User stories
  - Pair programming
  - Continuous integration
  - Test Driven Development
  - Customer on site
  - Small releases
- ✓ They are organized as rules for
  - Planning and Managing
  - Designing, Coding and Testing

source: http://www.extremeprogramming.org/rules.html

Pair programming

- ✓ Each pair includes two roles
  - *”The driver", focuses* on writing syntactically correct code
  - *”The navigator”*, considers how the code fits into the overarching design.
- ✓ Incremental development requires discipline and pair-programming is one technique for achieving that.
- ✓ Development will be less sensitive for external disturbances.

- ✓ Most important! Requires that individuals collaborate and communicate.

Continuous Integration – The “Integrate often rule”

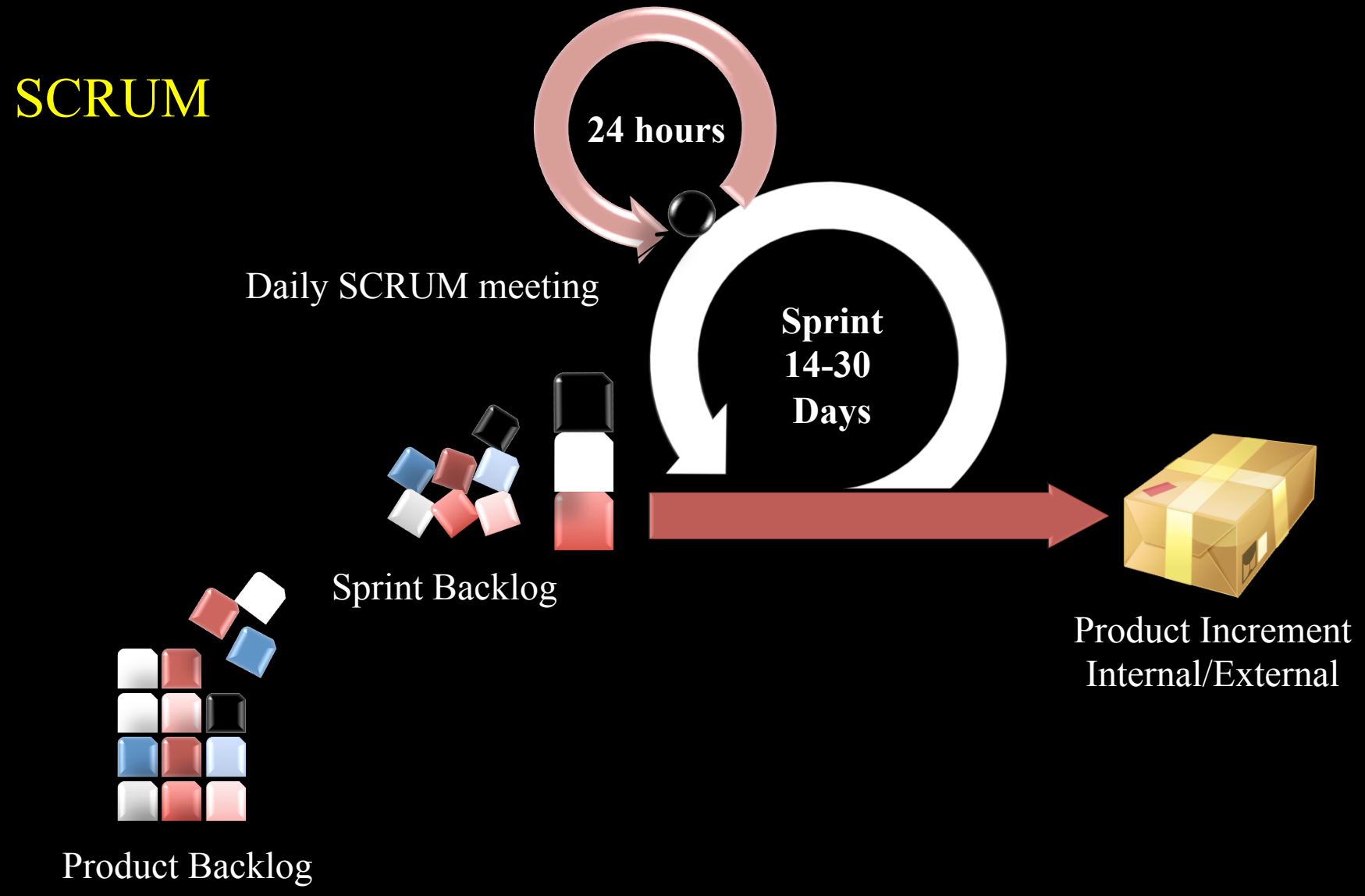
- ✓ The goal for continuous integration is that all code should be deployable to a customer.
- ✓ The rationale is that everything that is produced should fit the larger context, even though all functionality is not there yet.

- ✓ CI forces you to integrate smaller pieces, which makes life a lot easier.
- ✓ Requires automation

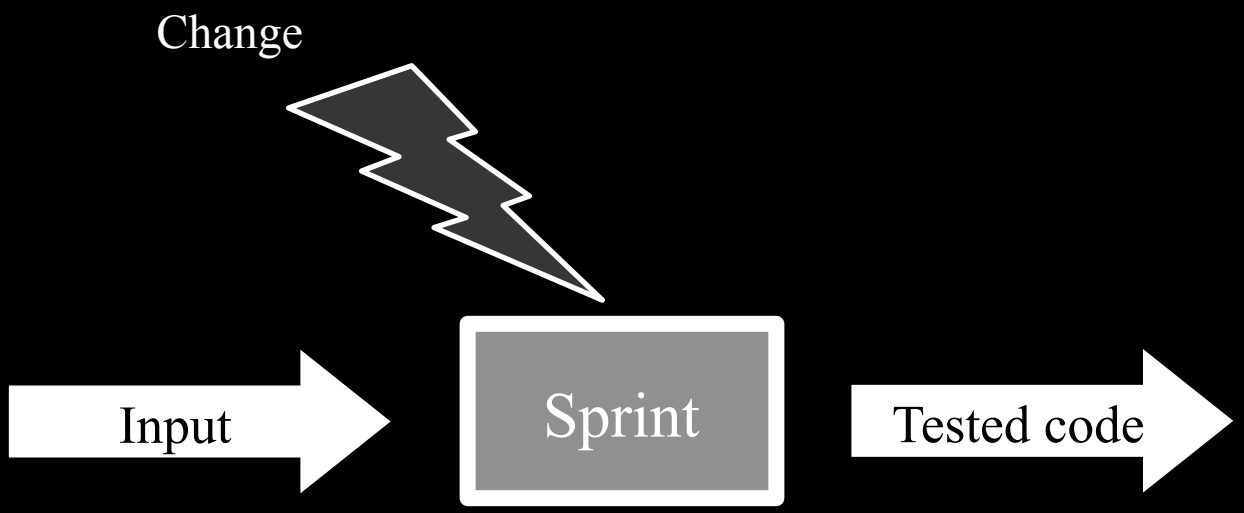
SCRUM

- ✓ An agile process first mentioned in 1996
- ✓ Most important principle is “Self-organizing teams”, that is shared leadership
- ✓ Product evolves in a number of sprints.
- ✓ The requirements are collected in a list, the product backlog
- ✓ The process does not prescribe any techniques or methods for the ”development”

SCRUM


24 hours
Daily SCRUM meeting
Sprint
14-30
Days
Sprint Backlog
Product Increment
Internal/External
Product Backlog

Principle: No changes during a sprint

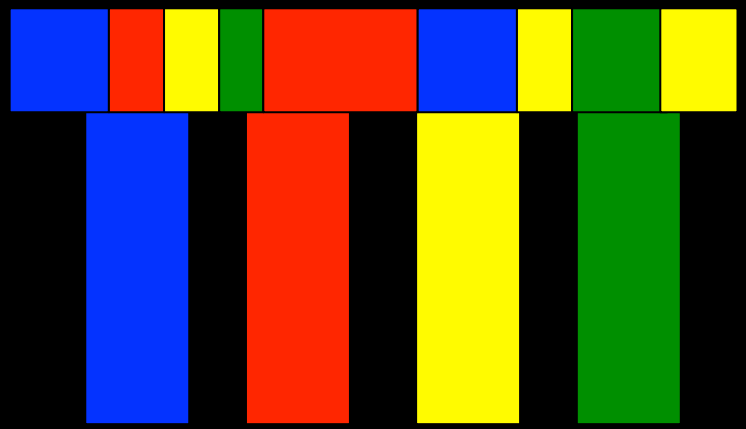


"A sprint should continue until it is impossible to keep change requests away"

Principle: “Cross-functional” Development Team

- ✓ Small group of self-organized individuals
- ✓ Should be able to produce executable code that the customer evaluates
- ✓ Covers all methods and technologies used.
- ✓ The team can only change in-between two sprints
- ✓ All members work full time in the project

- ✓ Robustness
- ✓ Flexibility



SCRUM Roles and Activities

- ✓ Roles
  - Product Owner – owner of the product vision
  - Scrum Master – assist the team in their application of SCRUM
  - Development team – builds the product
- ✓ Artifacts
  - Product increment – a finished part of a product.
  - Product backlog – a prioritized list of ideas for a product
  - Sprint backlog – a detailed plan for the development during the next sprint.
- ✓ Ceremonies
  - Sprint Planning – planning in the beginning of a sprint
  - Sprint Review – ”demonstration” of sprint result
  - Sprint Retrospective – ”sprint” post-mortem, improvement meeting
  - Daily Scrum Meeting – short daily meeting with team and scrum master

Product Backlog

- ✓ A list of all desired work on the project
- ✓ Contains a combination of
  - – “story”- based work, cmp. to. requirements (“let the user search the product database”)
  - – Task based work (“improve system logging functionality”)
- ✓ Prioritized by the Product Owner

- ✓ As a <type of user>, I want <some goal> so that <some reason>.

- ✓ May also include “Shores” (Swedish:sysslor)
  - – ”Install tools”
  - – Backup the development database

The two step planning meeting

- ✓ 1st step – with the product owner
  - Create a Product backlog
  - Decide which goals you have for the sprint.
  - Participants in the 1st step are
    - Product Owner,
    - Scrum Master,
    - Development team
- ✓ 2nd step – development team internal
  - Create a sprint backlog
  - Participants
    - Scrum Master
    - Development team

Today's takeaways

- ✓ Iterative & Incremental decomposition of projects
- ✓ Process models – "shades of grey"
  - – You do the same things!
  - – However some models are much more prescriptive!
- ✓ Kanban – No roles, no iterations, just a flow
- ✓ Scrum – Some roles, some artifacts, some ceremonies
- ✓ OpenUP – Prescriptive, provides guidance in most situations

✓ Agile requires a different skill set and experience!

Next lecture

- ✓ Planning
  - Planning a project, plan driven projects
  - Agile planning, value driven planning
- ✓ Managing projects
  - Project Manager, or
  - Coach