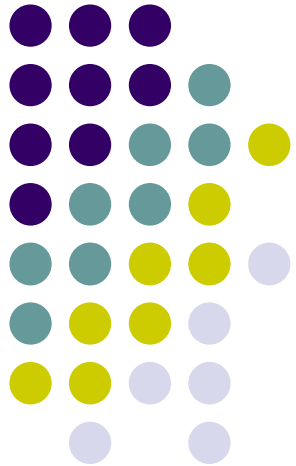
INTRODUCTION TO SOFTWARE ENGINEERING

Part I

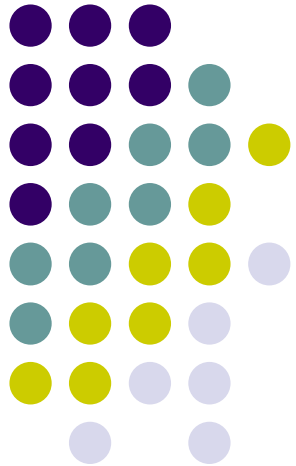


Overview of Software Engineering

- What is software engineering?
- Software process models
- Software project management
  - Project schedule
  - Project personnel
  - Quality management
  - Software configuration management
  - …

INTRODUCTION TO SOFTWARE ENGINEERING

Part I.2 – Software Process Models



Content

- The Meaning of Process
- Software Process Models (Software Development Process Models)

The Meaning of Process

- A process: a series of steps involving *activities*, *constrains*, and *resources* that produce an intended output of some kind
- A process involves a set of *tools* and *techniques*

The Meaning of Process
Process Characteristics

- Prescribes all major process *activities*
- Uses *resources*, subject to set of *constraints* (such as schedule)
- Produces intermediate and final products
- May be composed of *sub-processes* with hierarchy or links
- Each process activity has *entry and exit* criteria
- Activities are organized in *sequence*, so timing is clear
- Each process *guiding principles*, including goals of each activity
- Constraints may apply to an activity, resource or product

The Meaning of Process
The Importance of Processes

- Impose consistency and structure on a set of activities
- Guide us to understand, control, examine, and improve the activities
- Enable us to capture our experiences and pass them along to others

Software Process Models

- Modeling a process
  - The description of approach to produce a software as it be done in practice.
  - The rules of software development process should progress.

Software Process Models
Reasons for Modeling a Process

- To form a common understanding
- To find inconsistencies, redundancies, omissions
- To find and evaluate appropriate activities for reaching process goals
- To tailor a general process for a particular situation in which it will be used

Software Process Models
Software Life Cycle

- When a process involves building a software, the process may be referred to as **software life cycle**
  - Requirements analysis and definition
  - System (architecture) design
  - Program (detailed/procedural) design
  - Writing programs (coding/implementation)
  - Testing: unit, integration, system
  - System delivery (deployment)

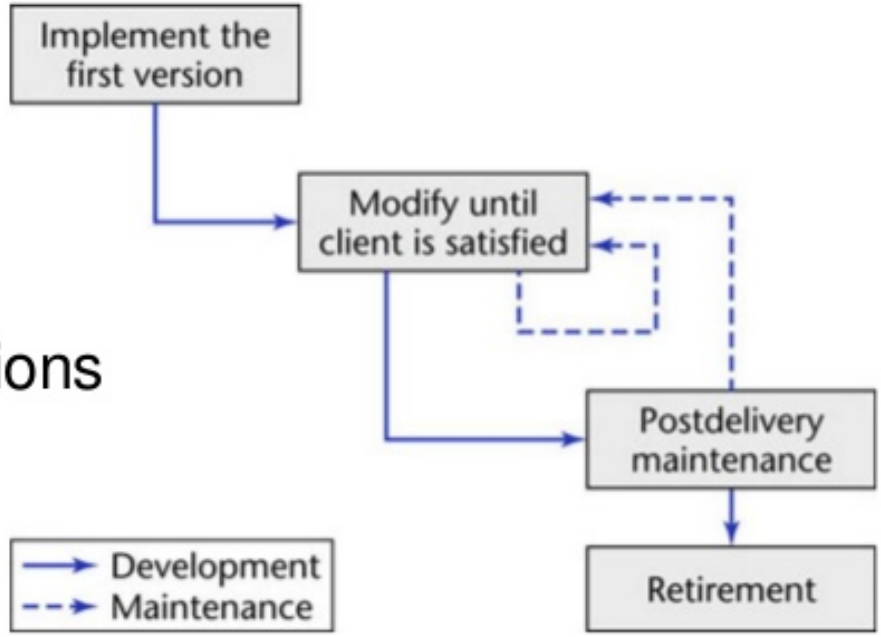
  - Maintenance

Software Process Models

- Code-and-Fix model
- Waterfall model
- V model
- Prototyping model
- Increments and Iterations model
- Spiral model
- Rapid Application Development
- Agile Methods (Extreme Programming, Scrum)
- RUP
- …

Code-and-Fix model

➢ **Code-and-Fix Life-Cycle Model**



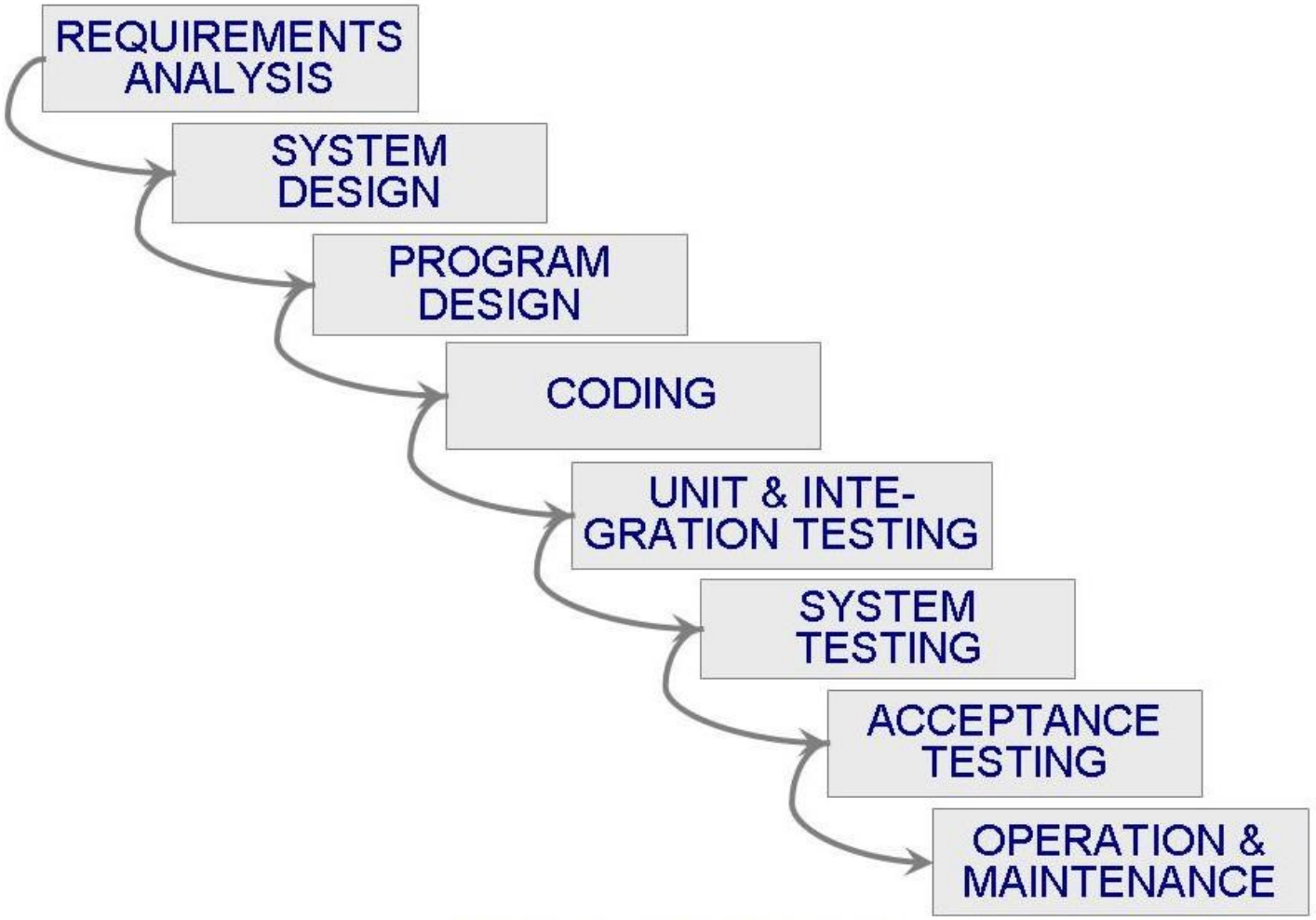
- No design
- No specifications

The easiest way to develop software
The most expensive way for maintenance
(i.e., maintenance nightmare)

Waterfall Model

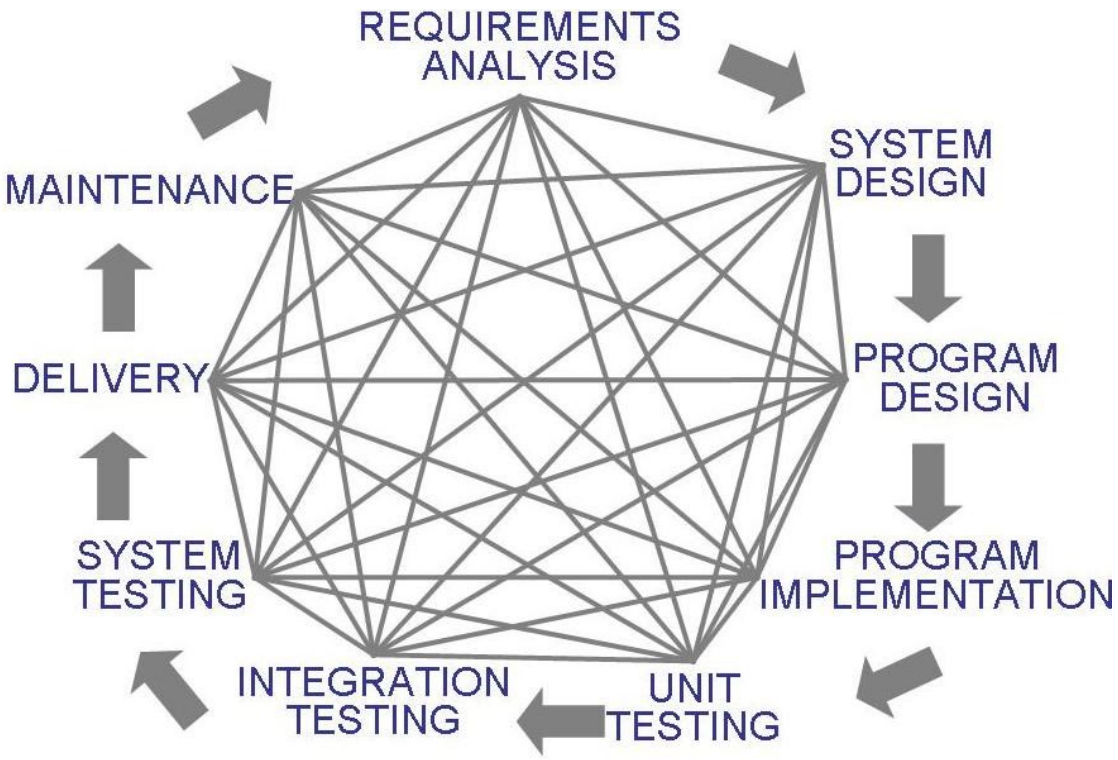
- One of the first process development models proposed
- Works for well understood problems with minimal or no changes in the requirements
- Simple and easy to explain to customers
- It presents
  - a very high-level view of the development process
  - sequence of process activities
- Each major phase is marked by milestones and deliverables (artifacts)

Waterfall Model



Waterfall Model

- There is no iteration in waterfall model
- Most software developments apply a great many iterations



Software Process Models
Drawbacks of The Waterfall Model

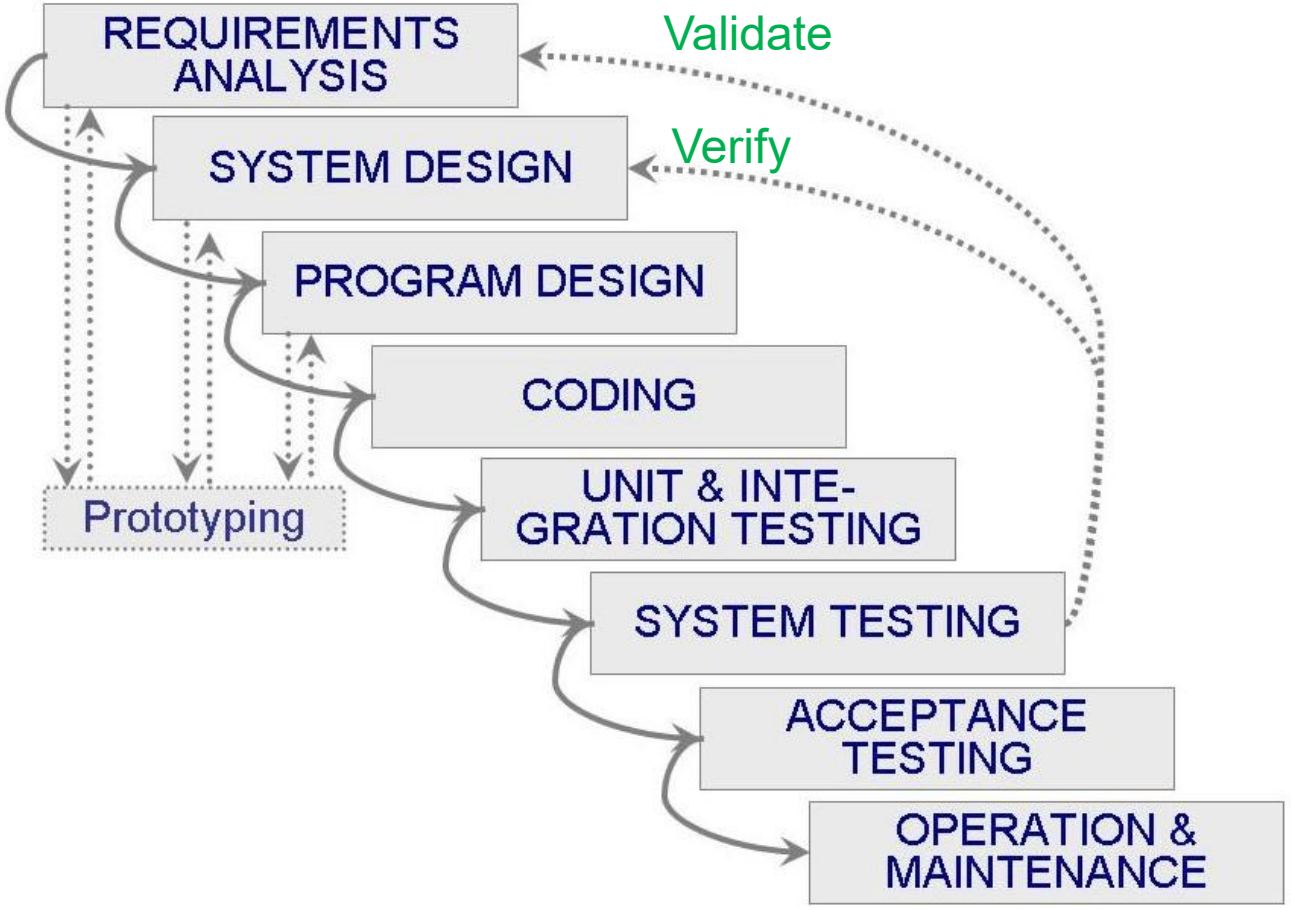
- Provides no guidance how to handle changes to products and activities during development (assumes requirements can be frozen)
- Views software development as manufacturing process rather than as creative process
- There is no iterative activities that lead to creating a final product
- Long wait before a final product

Waterfall Model with Prototype

- A prototype is a partially developed product
- Prototyping helps
  - developers assess alternative design strategies (design prototype)
  - users understand what the system will be like (user interface prototype)
- Prototyping is useful for verification and validation

Waterfall Model with Prototype

- Waterfall model with prototyping



Validation & Verification

- Validation ensures that the system has implemented all of the requirements, so that each system function can be traced back to a particular requirement in the specification.
- Verification ensures that each function works correctly.

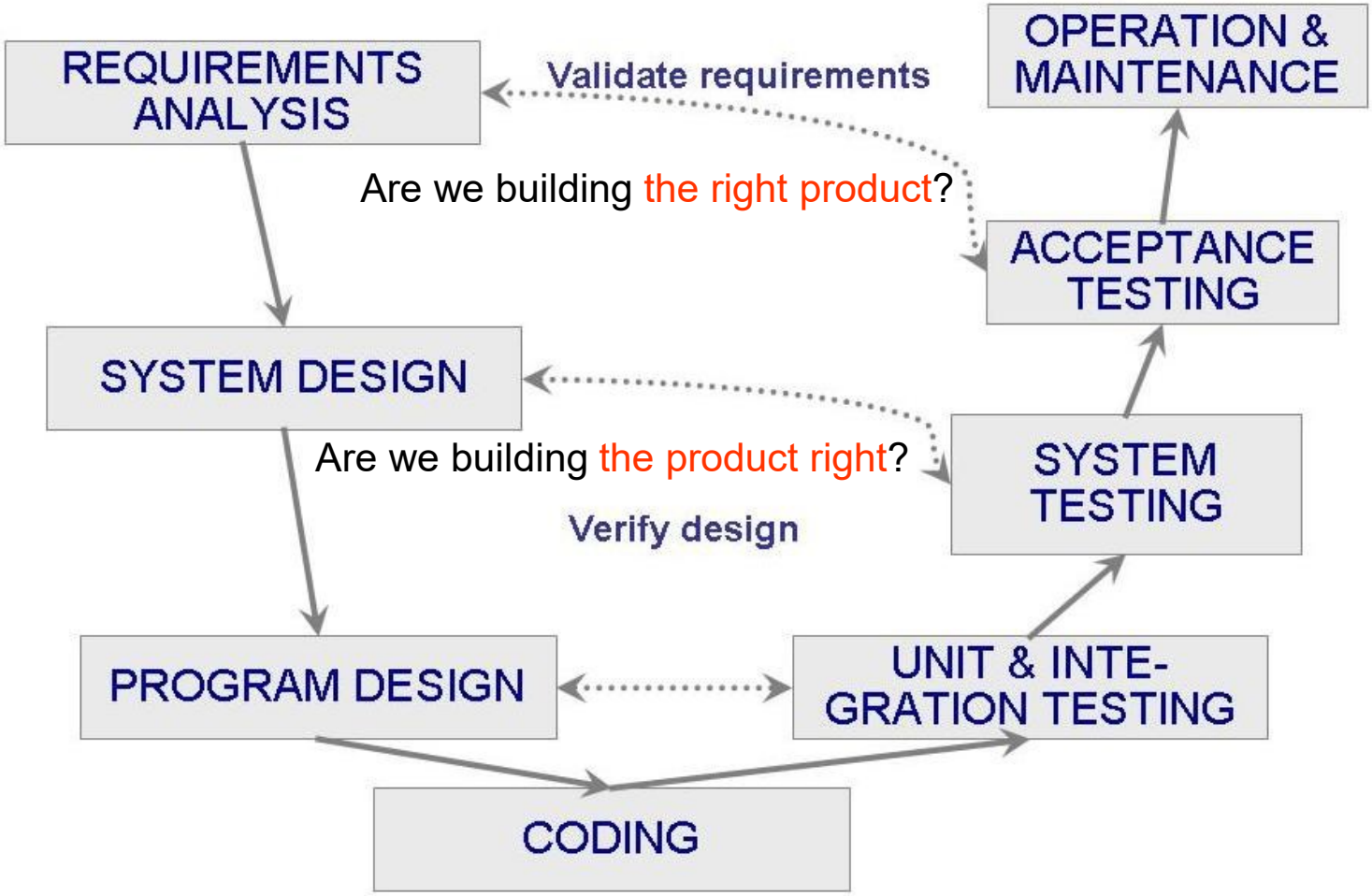
➡ *Validation* makes sure that the developer is building the right product (according to the specification). *Verification* checks the quality of the implementation.

- Capability Maturity Model (CMMI-SW v1.1),
  - Validation: The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements. [IEEE-STD-610]
  - Verification: The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. [IEEE-STD-610]

V Model

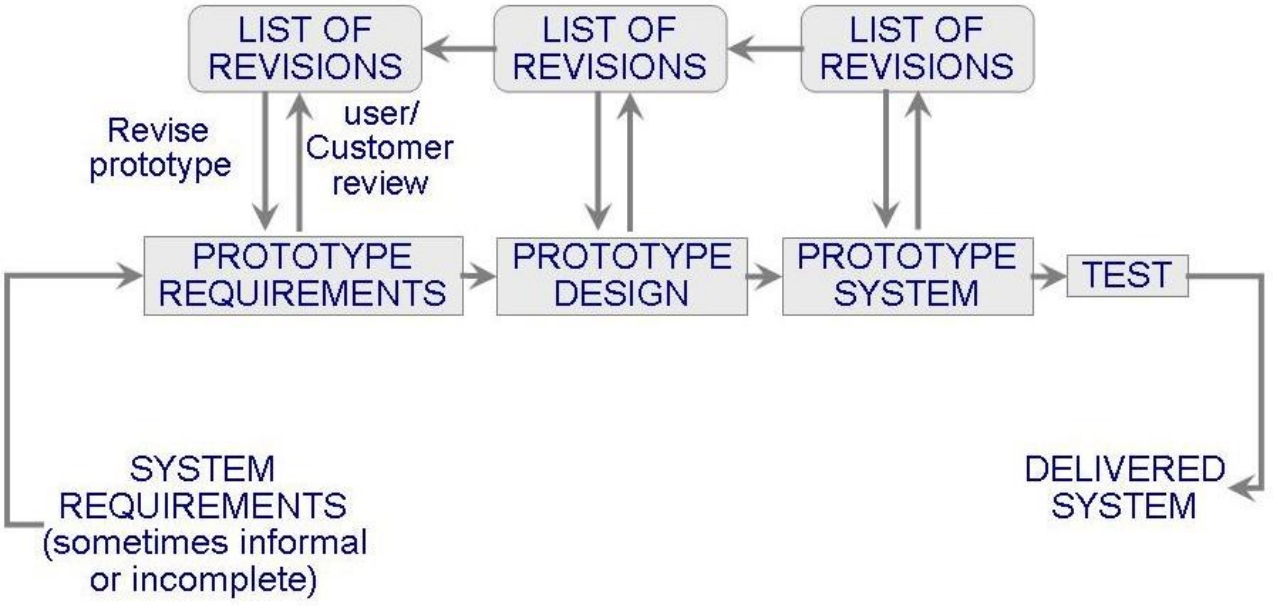
- A variation of the waterfall model
- Uses unit testing to verify procedural design
- Uses integration testing to verify architectural (system) design
- Uses acceptance testing to validate the requirements
- If problems are found during verification and validation, the left side of the V can be re-executed before testing on the right side is re-enacted

V Model


REQUIREMENTS ANALYSIS
Validate requirements
OPERATION & MAINTENANCE
Are we building the right product?
ACCEPTANCE TESTING
SYSTEM DESIGN
Are we building the product right?
SYSTEM TESTING
Verify design
PROGRAM DESIGN
UNIT & INTE-GRATION TESTING
CODING

Prototyping Model

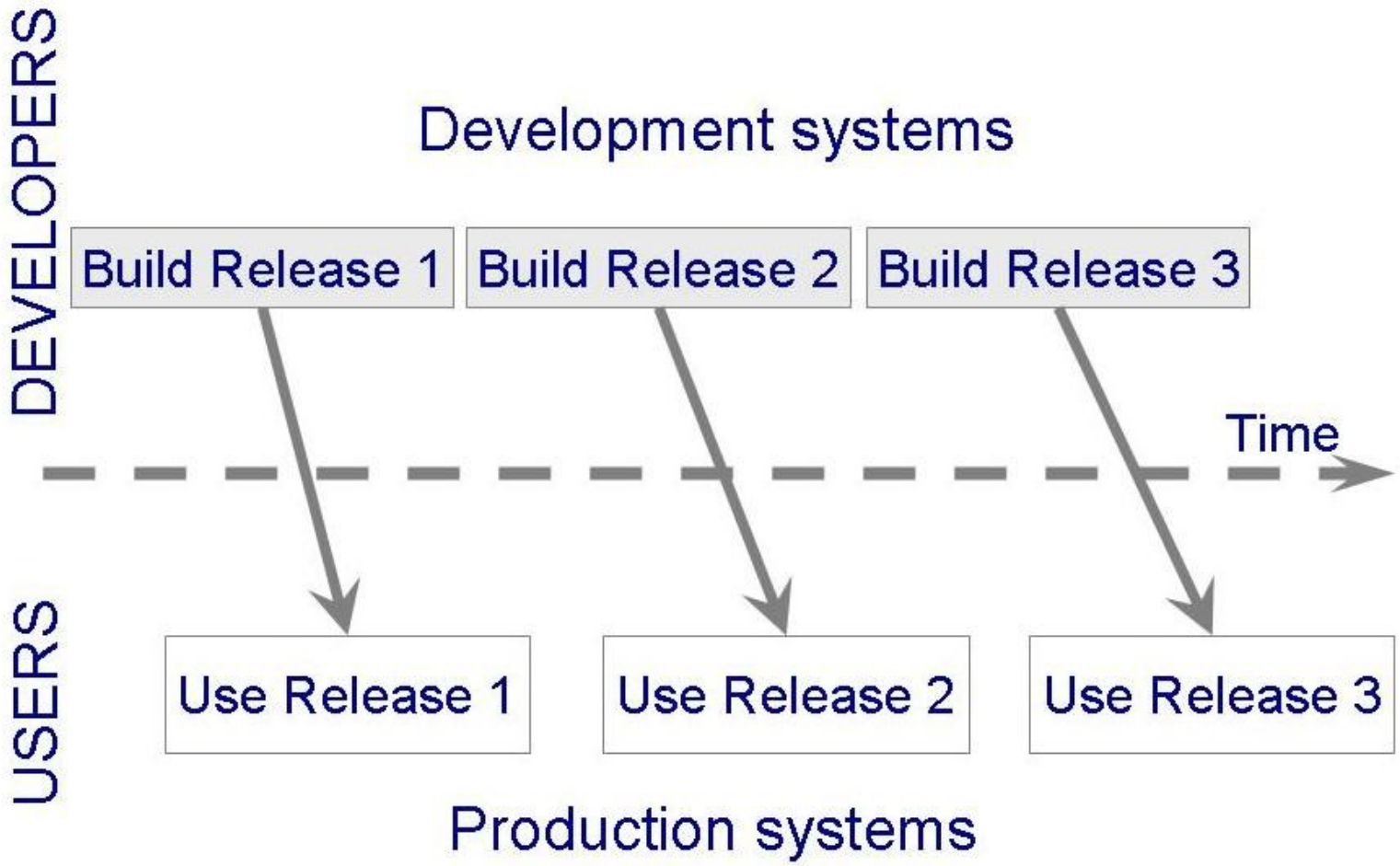
- Allows repeated investigation of the requirements or design
- Reduces risk and uncertainty in the development



Phased Development Models: Increments and Iterations

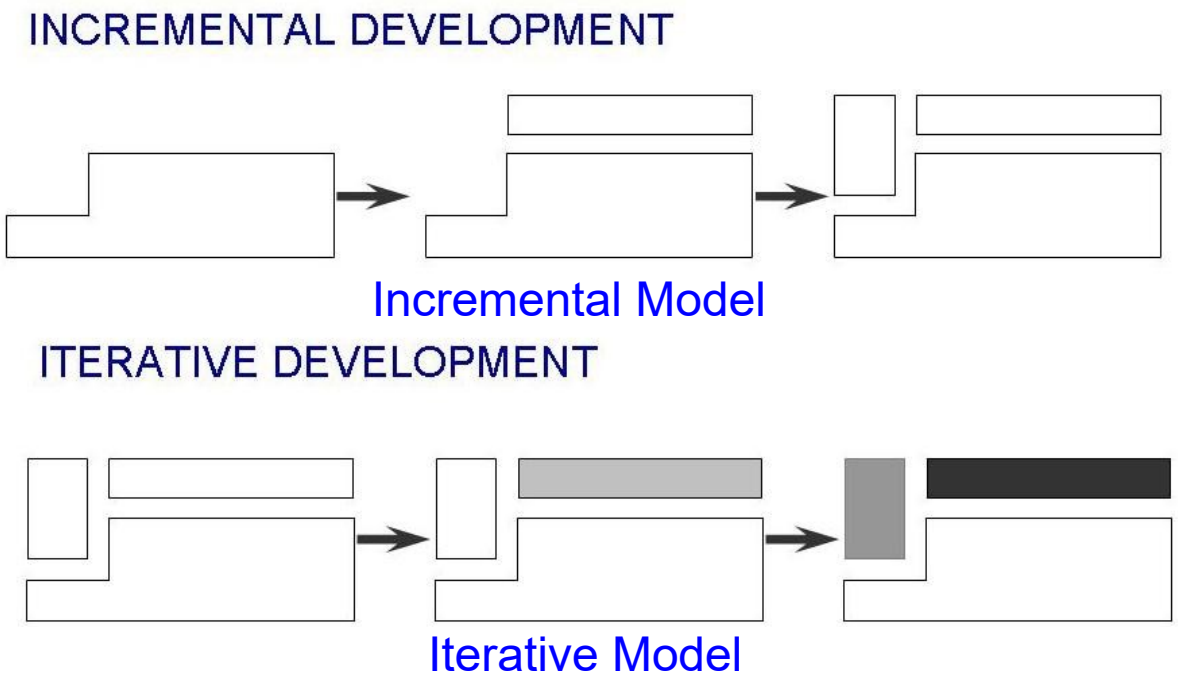
- Shorter cycle time
- System delivered in pieces
  - enables customers to have some functionality while the rest is being developed
- Allows two systems functioning in parallel
  - the production (or operational) system (release *n*): currently being used
  - the development system (release *n+1*): the next version

Phased Development Models: Increments and Iterations



Phased Development Models: Increments and Iterations

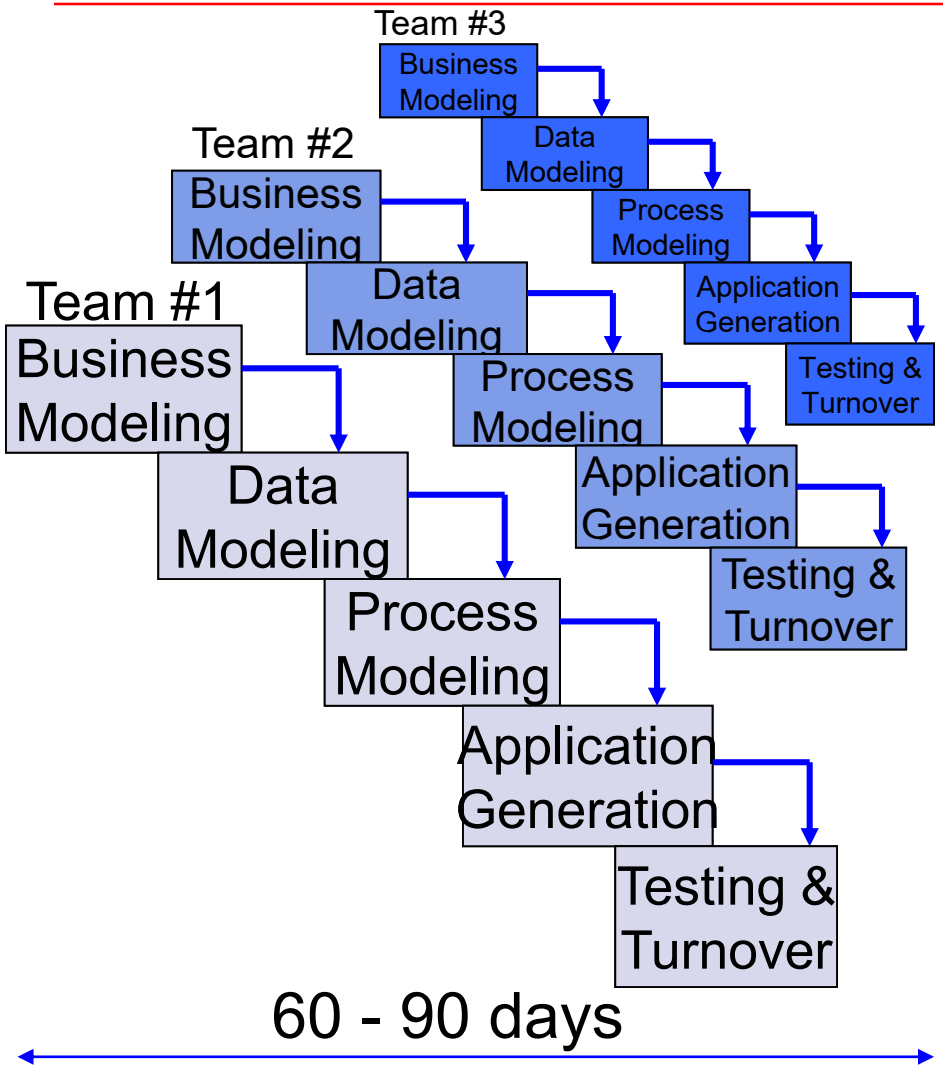
- **Incremental development:** starts with small functional subsystem and adds functionality with each new release
- **Iterative development**: starts with full system, then changes functionality of each subsystem with each new release



Phased Development Models: Increments and Iterations

- Phased development (combination of iterative and incremental development) is desirable for several reasons
  - Training can begin early, even though some functions are missing
  - Markets can be created early for functionality that has never before been offered
  - Frequent releases allow developers to fix unanticipated problems globally and quickly
  - The development team can focus on different areas of expertise with different releases

Rapid Application Development (RAD)



- It is a type of incremental model, increases progressively within very short development cycle (60-90 days).
- Increases reusability of components.
- Involve several teams, each team is a RAD including phases: *Business modeling, Data modeling, Process modeling, Application generation, Testing and turnover*.

Rapid Application Development (RAD)

- Reduces the development time.
- Increases the reusability of components.
- Encourages the customer feedback.

Rapid Application Development (RAD)

- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD.
- Requires highly skilled evelopers/designers.
- High dependency on modeling skills.
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

Spiral Model

- Suggested by Boehm (1988)
- Combines development activities with risk management to minimize and control risks
- The model is presented as a spiral in which each iteration is represented by a circuit around four major activities
  - Plan
  - Determine goals, alternatives and constraints
  - Evaluate alternatives and risks
  - Develop and test

Spiral Model

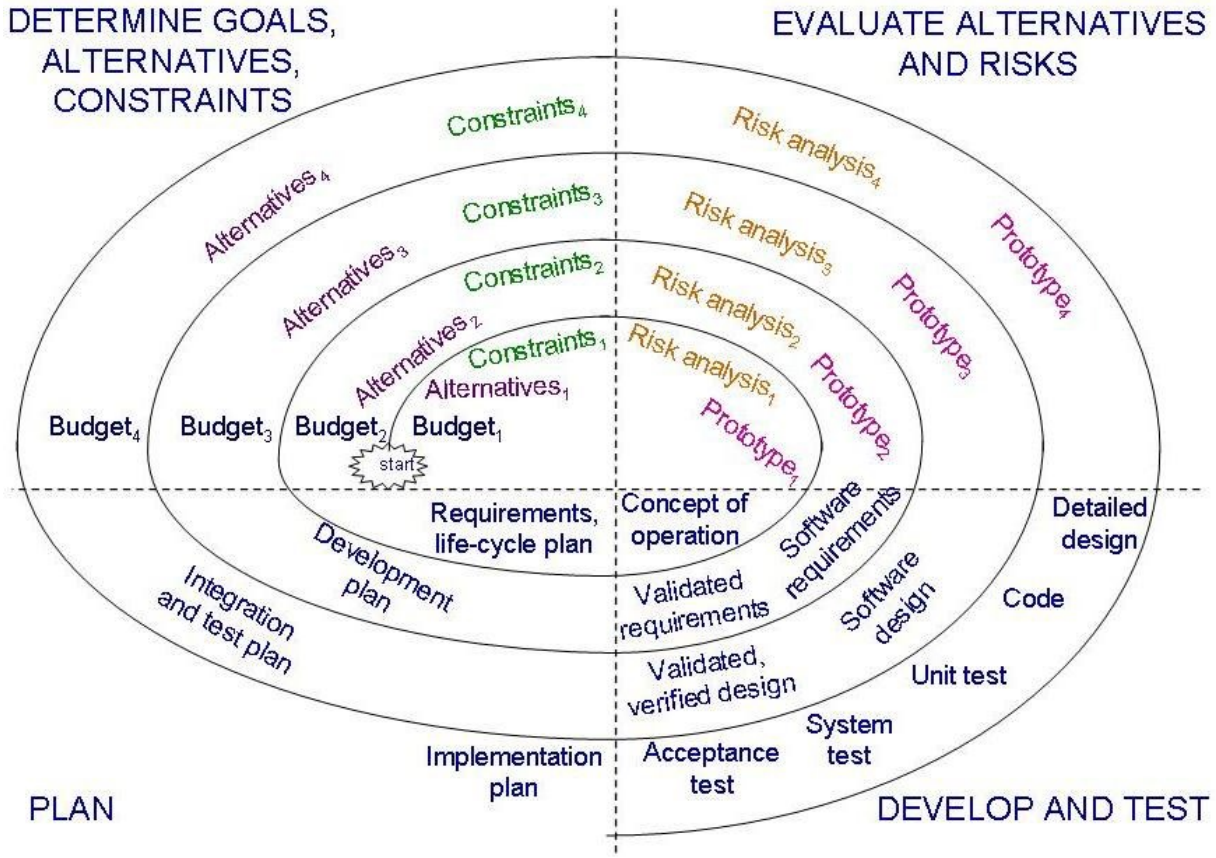
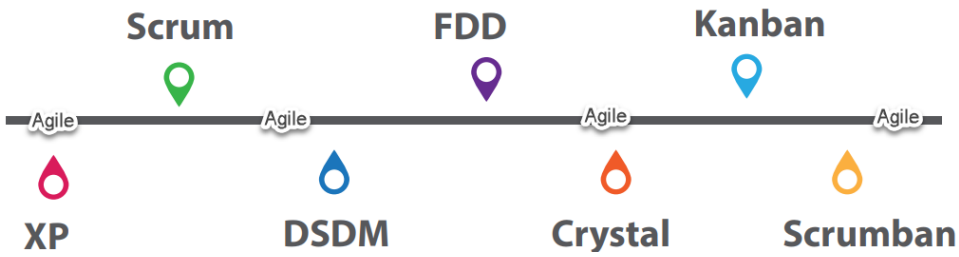


Figure 2.10 the spiral model.

Agile Methods

- Emphasis on flexibility in producing software quickly and capably
- Agile manifesto
  - Value individuals and interactions over process and tools
  - Prefer to invest time in producing working software rather than in producing comprehensive documentation
  - Focus on customer collaboration rather than contract negotiation
  - Concentrate on responding to change rather than on creating a plan and then following it

Agile Methods - Models



CHAOS RESOLUTION BY AGILE VERSUS WATERFALL (2015)

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

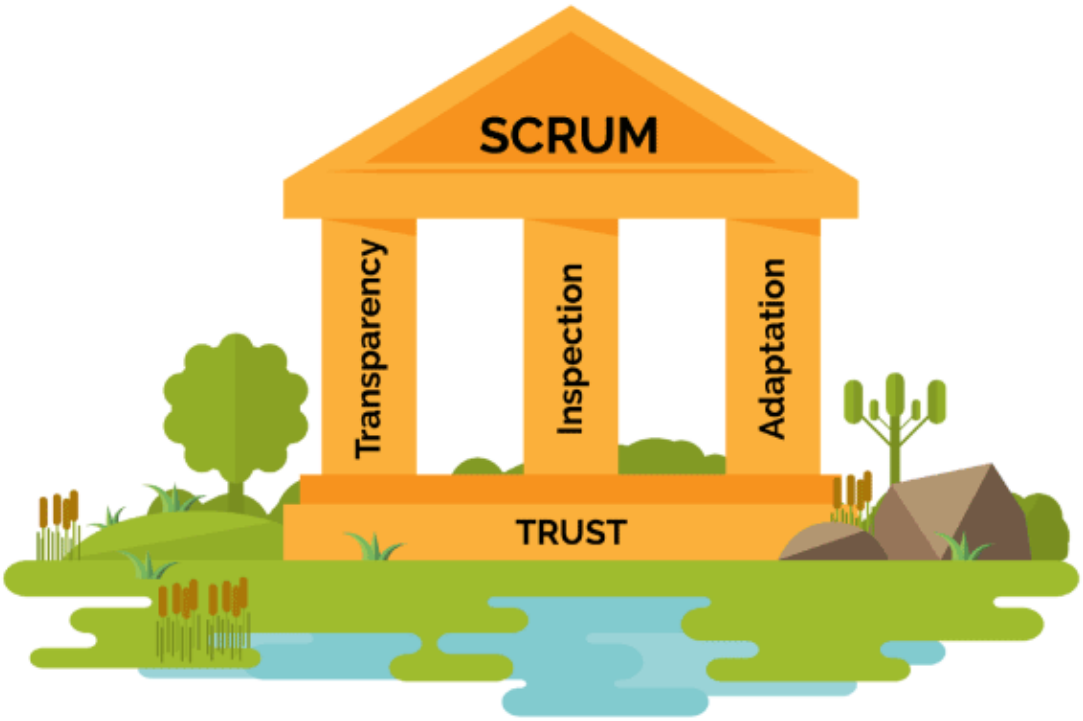
Agile Methods: Extreme Programming (XP)

- Emphasis on four characteristics of agility
  - *Communication*: continual interchange between customers and developers
  - *Simplicity*: select the simplest design or implementation
  - *Courage*: commitment to delivering functionality early and often
  - *Feedback*: loops built into the various activities during the development process

Agile Methods: Extreme Programming (XP)

- These characteristics are embedded in what are known as the twelve facets of XP:
  - The planning game
  - Small releases
  - Metaphor (common vision of how the system will operate)
  - Simple design
  - Writing tests first,
  - Refactoring
  - Pair programming
  - Collective ownership
  - Continuous integration
  - Sustainable pace
  - On-site customer
  - Coding standards

Agile Methods: SCRUM



COURAGE
Scrum Team members have courage to do the right thing and work on tough problems

FOCUS
Everyone focuses on the work of the Sprint and the goals of the Scrum Team

COMMITMENT
People personally commit to achieving the goals of the Scrum Team

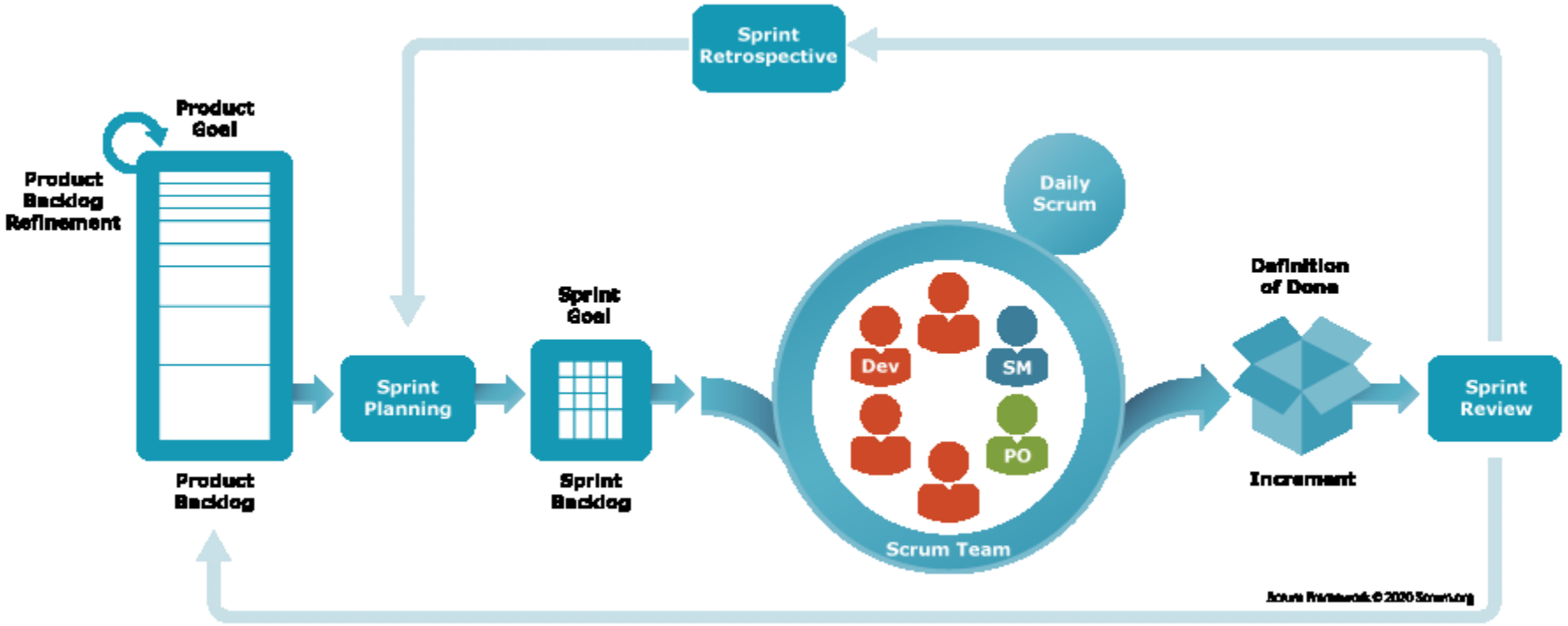
RESPECT
Scrum Team members respect each other to be capable, independent people

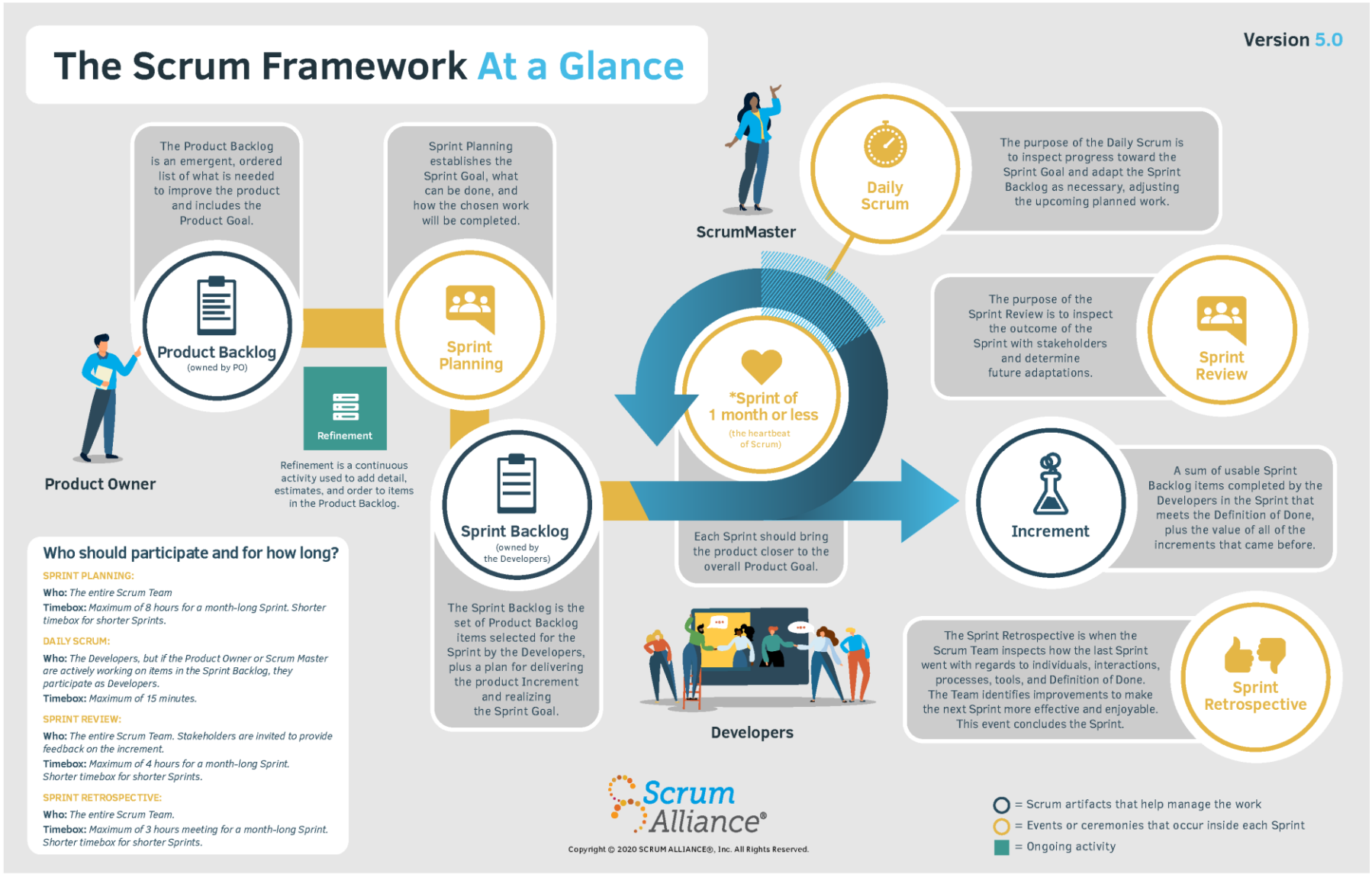
OPENNESS
The Scrum Team and its stakeholders agree to be open about all the work and the challenges with performing the work

Credit: ABN AMRO Bank N.V.

https://www.scrum.org/resources/what-scrum-module

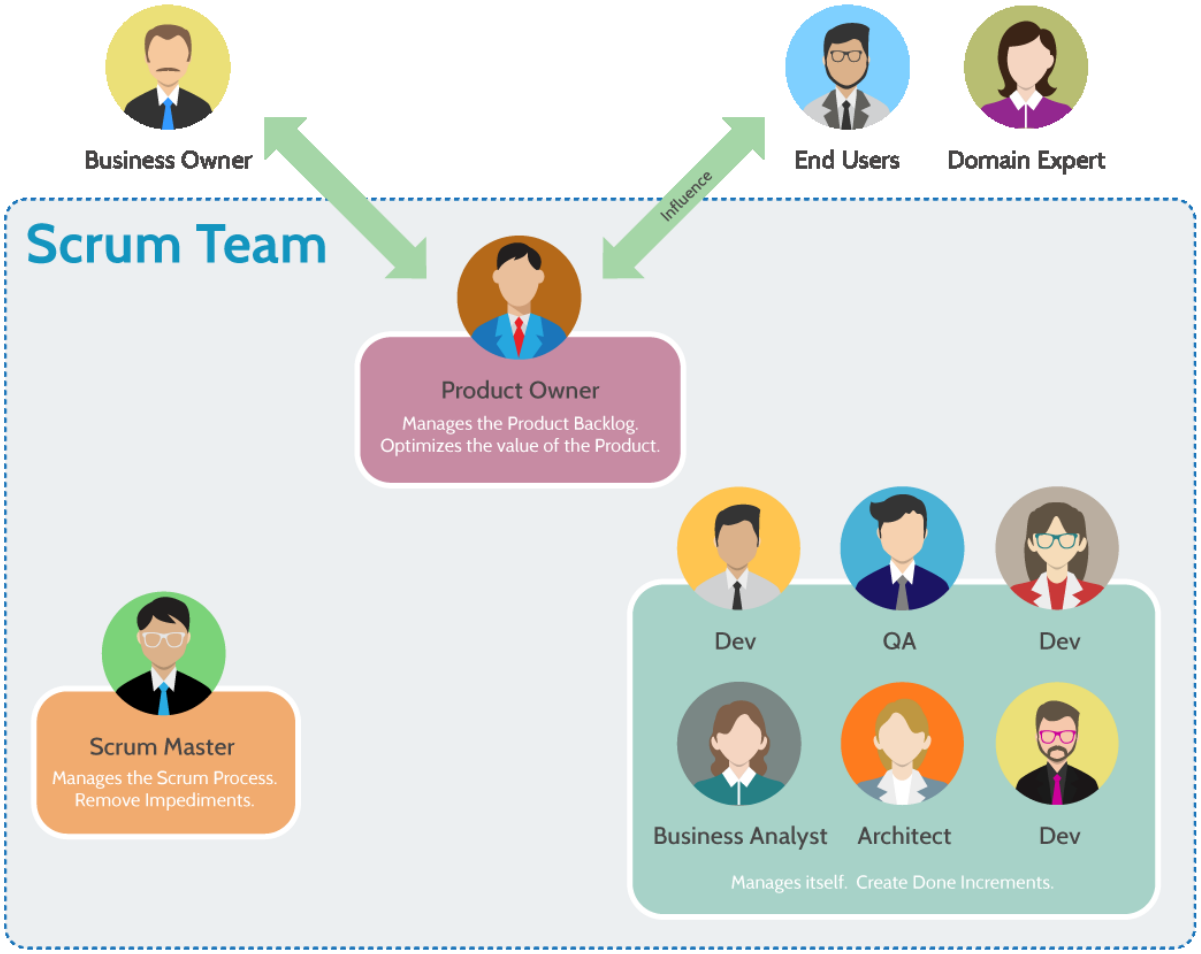
Agile Methods: SCRUM





https://www.scrumalliance.org/about-scrum

Agile Methods: SCRUM

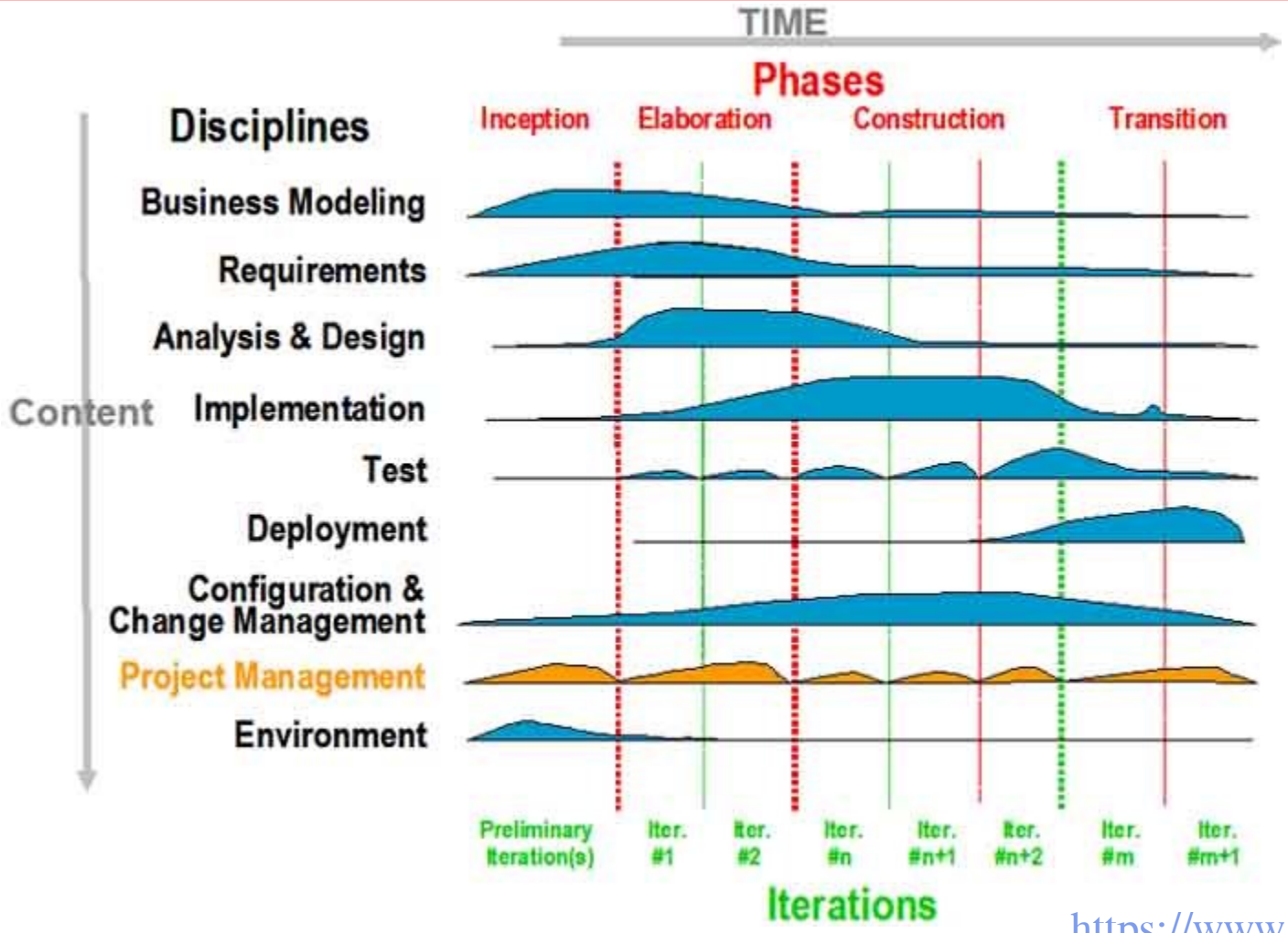


https://www.visual-paradigm.com/scrum/what-is-scrum-team/

Agile Methods: SCRUM

- Scrum is a process framework that has been used to manage work on complex products since the early 1990s.
- The Scrum framework consists of Scrum Teams and their associated roles, events, artifacts, and rules. Each component within the framework serves a specific purpose and is essential to Scrum's success and usage.

Rational Unified Process - RUP



https://www.ibm.com

Rational Unified Process - RUP

- **The dynamic aspect of process (Phases and Iterations)**
  - Inception: establish the business case for the system and delimit the project scope. The business case includes success criteria, risk assessment, and estimate of the resources needed, and a phase plan showing dates of major milestones.
  - Elaboration: analyze the problem domain, establish a sound architectural foundation, develop the project plan, and eliminate the highest risk elements of the project.
  - Construction: all remaining components and application features are developed and integrated into the product, and all features are thoroughly tested.
  - Transition: is to transition the software product to the user community.

Rational Unified Process - RUP

- **The static aspect of process (workflows)**

  A process describes who is doing what, how, and when. The RUP is represented using four primary modeling elements:

  - Workers, the 'who'
  - Activities, the 'how'
  - Artifacts, the 'what'
  - Workflows, the 'when'. There are nine core process workflows in the RUP, which represent a partitioning of all workers and activities into logical groupings

Q&A

Homework

- What software would you like to develop? Why?
- Which model will your group use to develop that software? Why?
- What functions that software should have?
- Who will use that software? The software functions for each group of users?