Software Processes

Outline

- Software Process and Models
- Software Engineering Process Activities
- Agile software development
- Software process simulation

- The following slides are based on Chapter 2 and Chapter 3 of Sommerville's Software Engineering book, 10th edition.

Software Process

- A process consists of activities.
- Software processes involves feasibilty study, specification(requirements definition), software design, coding and testing (validation and verification) and evolution.
- To describe a process, one may include:
  - Process activites
  - Artifacts or Products
  - Roles or Responsibilities

Types of Software Processes

- Plan-driven processes
  - All activities are planned prior to commencing.
- Agile processes
  - The plan will be refined and enhanced to accommodate requirements' changes.
- In real world, plan-driven and agile methods are exercised together to make processes more practical.

Process/Model/Methodology

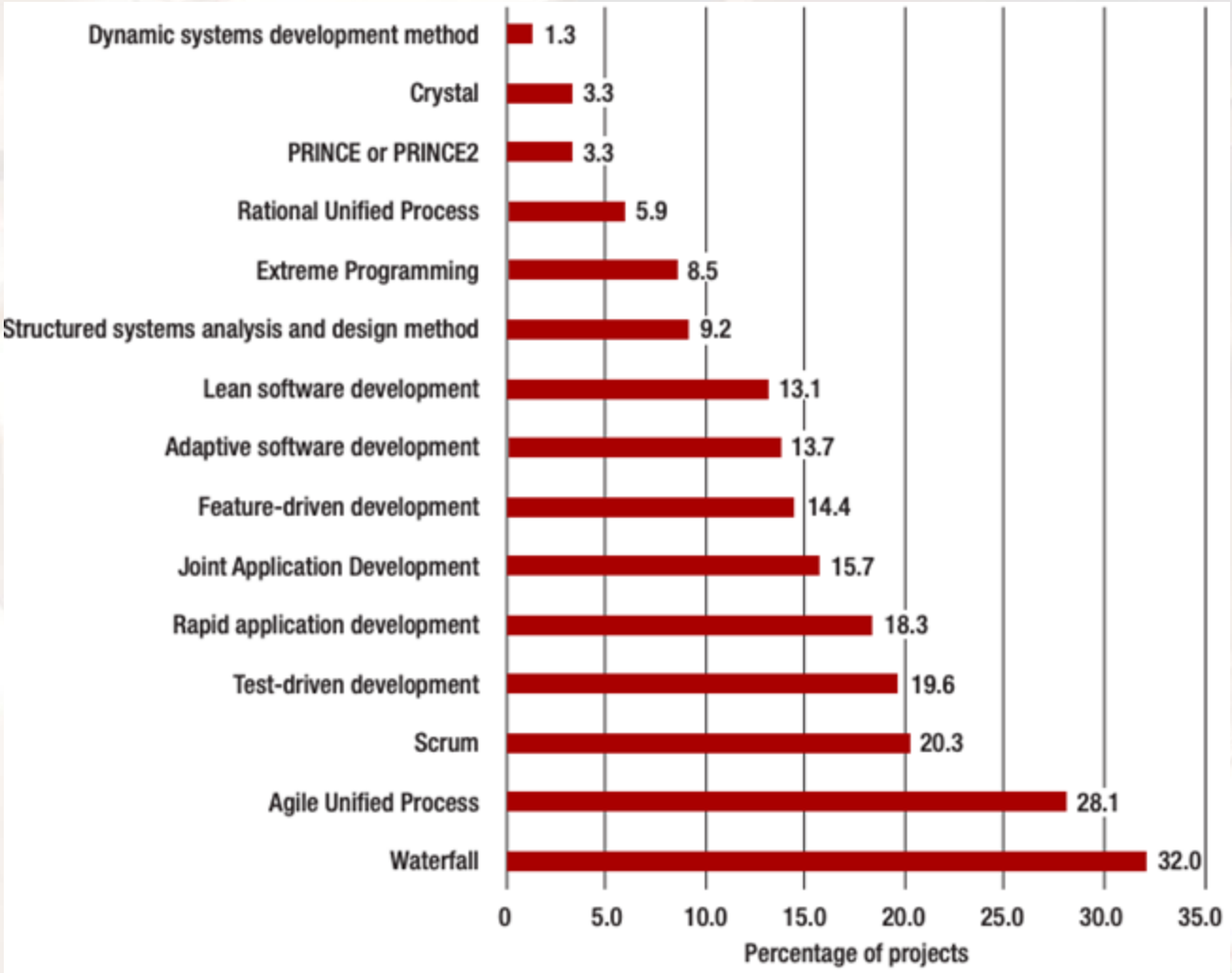
- Process
  - **Activities, actions and tasks** performed to create work products.
- Model
  - An abstract representation of the **software process**".
- Methodology
  - **Specification of how** to do each step.
- Plan-driven models
  - E.g. Waterfall model, Spiral model
- Agile methodologies
  - E.g. Extreme programming, Scrum

Models/Methodologies

- Waterfall model
- Iterative model
- V model
- Spiral model
- Extreme programming
- Incremental Model
- Rapid prototyping model
- Adaptive Software Development (ASD)
- Crystal Methodologies
- Dynamic System Development Method (DSDM)
- Feature Driven Development (FDD)
- Rational Unified Process (RUP)
- SCRUM

Bhuvaneswari, T., & Prabaharan, S. (2013). A survey on software development life cycle models. *International Journal of Computer Science and Mobile Computing*, *2*(5), 262-267.

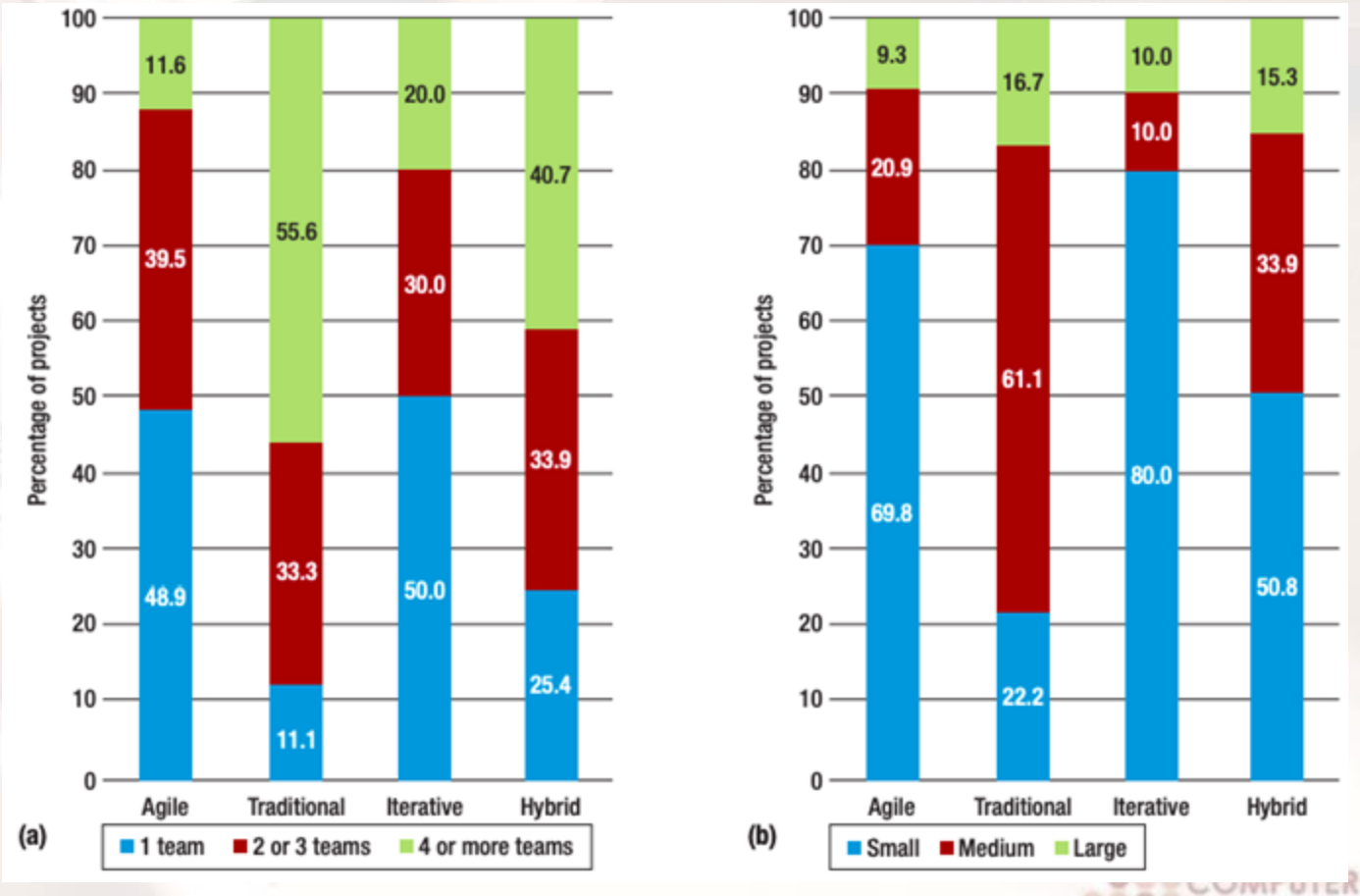
Choices of Methodologies



Vijayasarathy, L. R., & Butler, C. W. (2015). Choice of software development methodologies: Do organizational, project, and team characteristics matter?. *IEEE software*, *33*(5), 86-94.

Teams and Team Size

- Teams/Project
  - 1
  - 2-3
  - >= 4
- Team size
  - Small
  - Medium
  - Large



Vijayasarathy, L. R., & Butler, C. W. (2015). Choice of software development methodologies: Do organizational, project, and team characteristics matter?. *IEEE software*, *33*(5), 86-94.

Characteristics of projects following the four software development approaches.

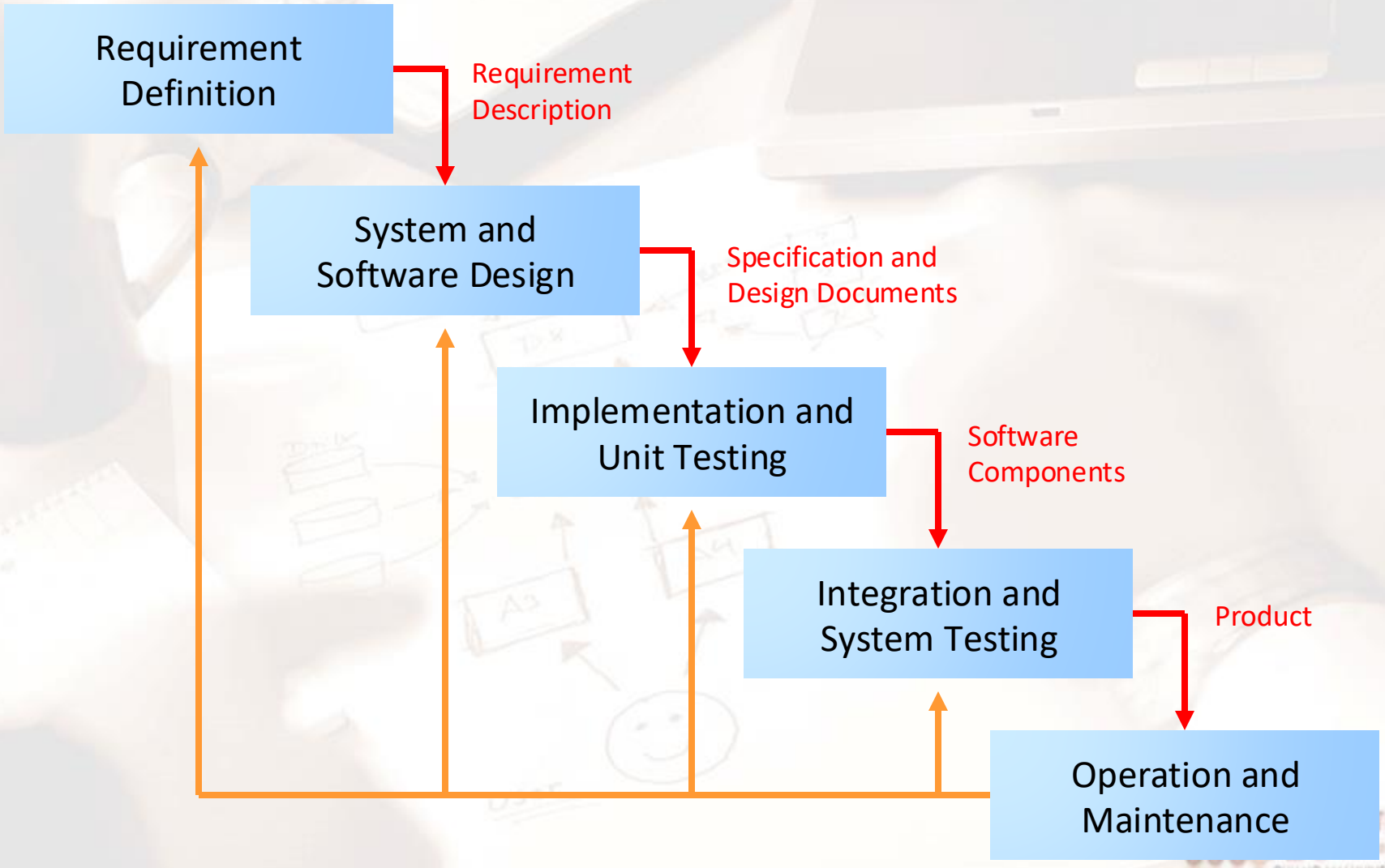
Approach	Characteristics		
	Organizational	Project	Team
Agile	Moderate revenue A small number of employees	Low budget Medium to high criticality	One team Small team
Traditional	High revenue A large number of employees	High budget High criticality	Multiple teams Medium team
Iterative	A small number of employees	Medium budget Medium to high criticality	One team Small team
Hybrid	Organization size unimportant	Medium budget High criticality	Small team

Vijayasarathy, L. R., & Butler, C. W. (2015). Choice of software development methodologies: Do organizational, project, and team characteristics matter?. *IEEE software*, *33*(5), 86-94.

Traditional Software Process

- Waterfall model
- Evolutionary model
- Incremental model
- Spiral model

Waterfall Model



Waterfall Model: Facts

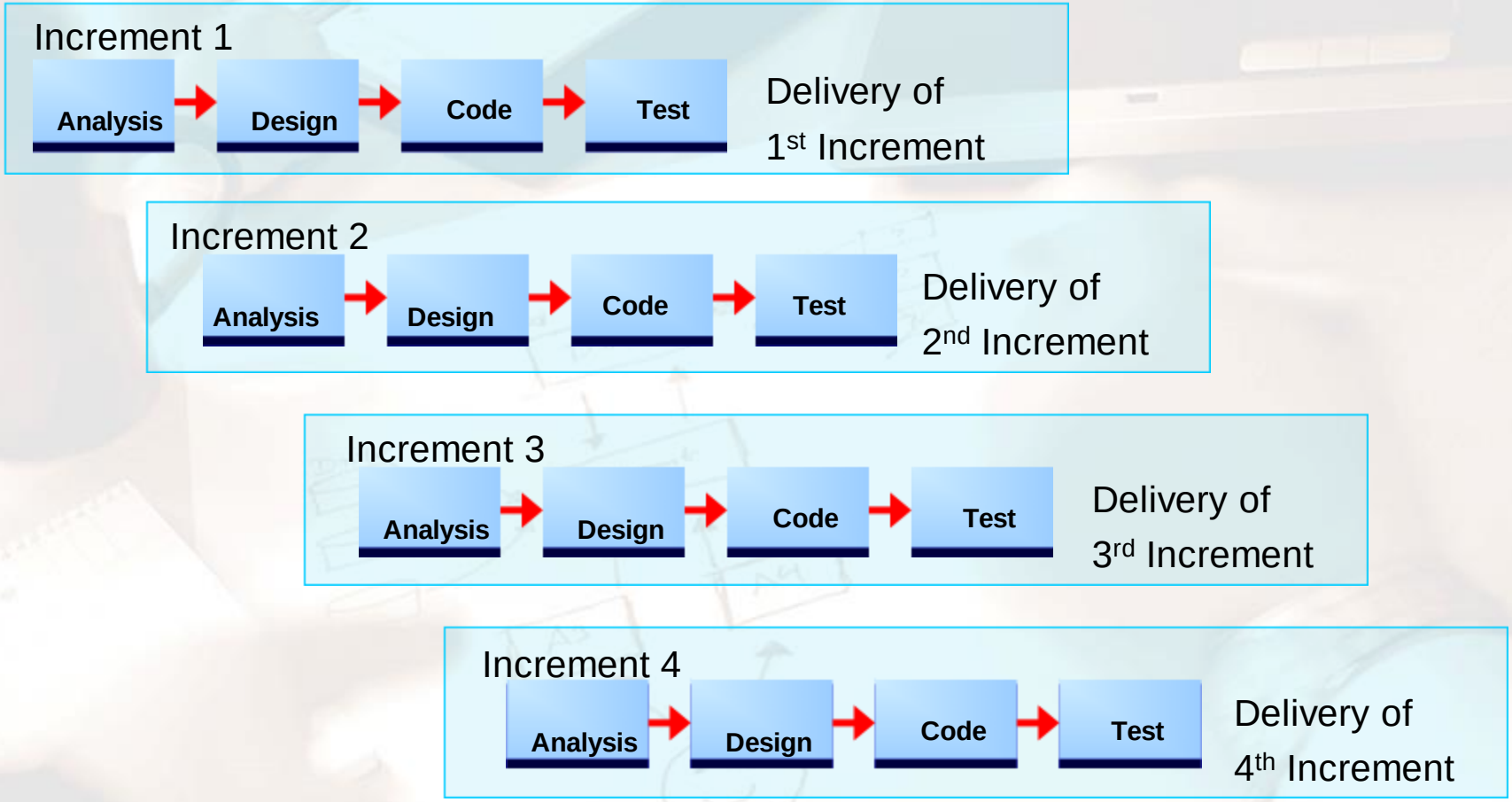
- **Ideally**, the phase must be finished before proceeding to the next phase.
- Changing in customer requirements can lead to unwanted outcomes. Therefore, requirements of software must be well-understood with limited changes for this model.
- **Fact**: Few business systems have stable requirements.

Waterfall Model: When to Use

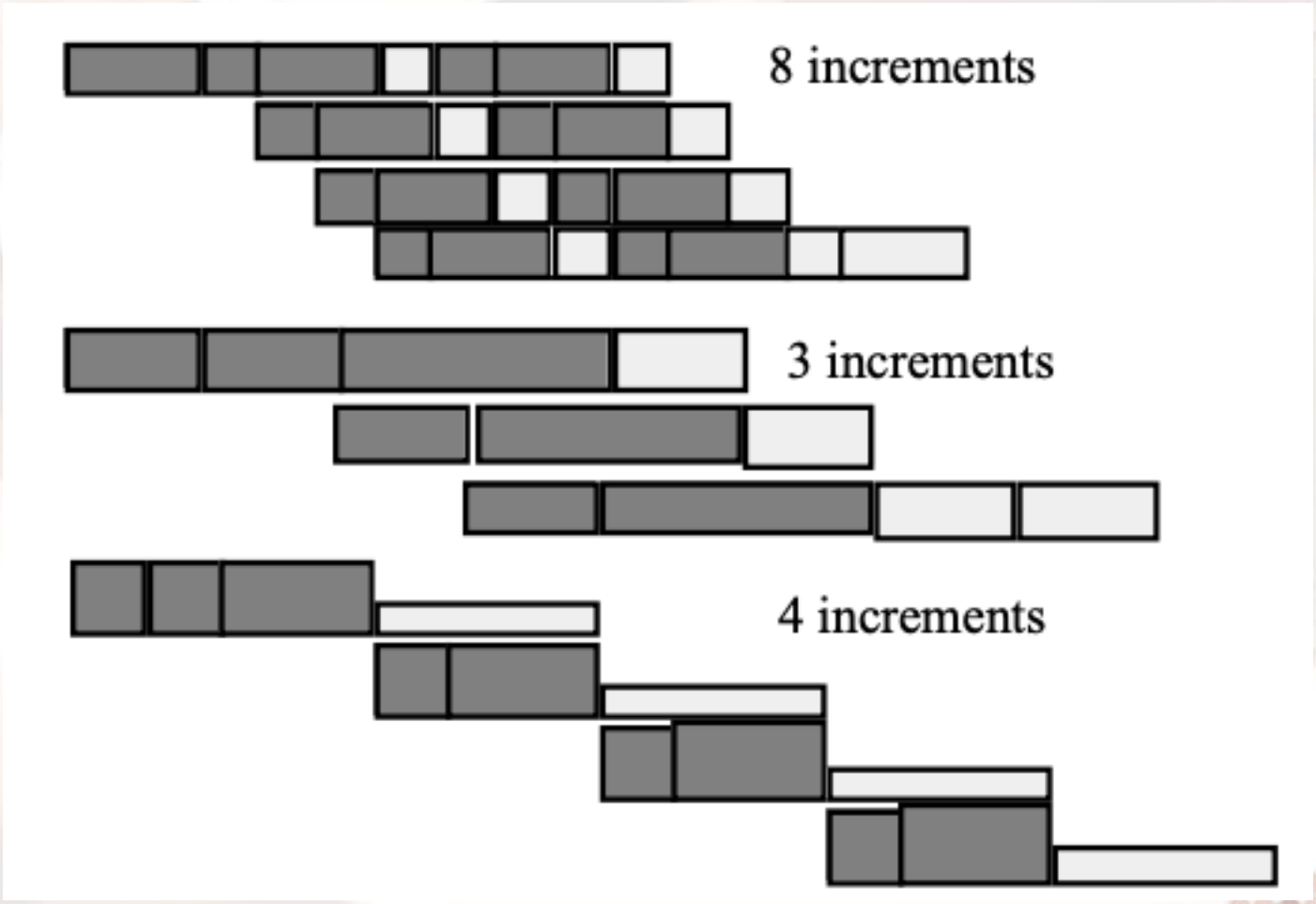
- The Waterfall model is suitable for large systems engineering projects.
- A system may be developed at several sites.
- The plan-driven model helps coordinate the work.

- **Review yourself (1)** : What are favorite and unfavorite factors of the Waterfall model? Why are they so?

Incremental Model



Incremental Model: Examples



- 2. Incremental development—terminology and guidelines, Even-André Karlsson, 2000

Incremental Development

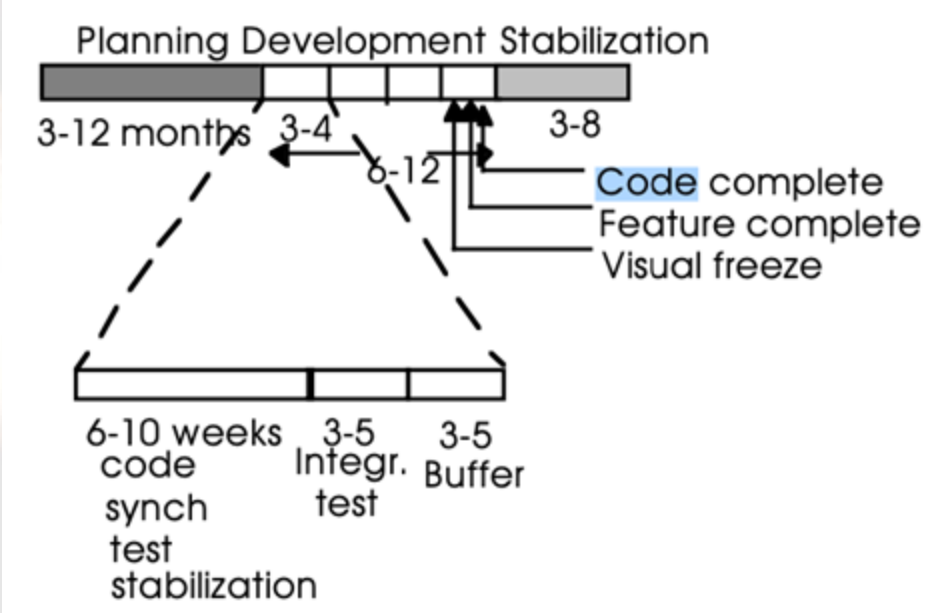
- The project with a greater number of increments has more opportunities for feedback[2] and accommodating changing customer requirements. However, requires better planning and coordination[2].
- Because of smaller increments, the amount of work, i.e., analysis and documentation, is much less when compared to the waterfall model.

Improvement

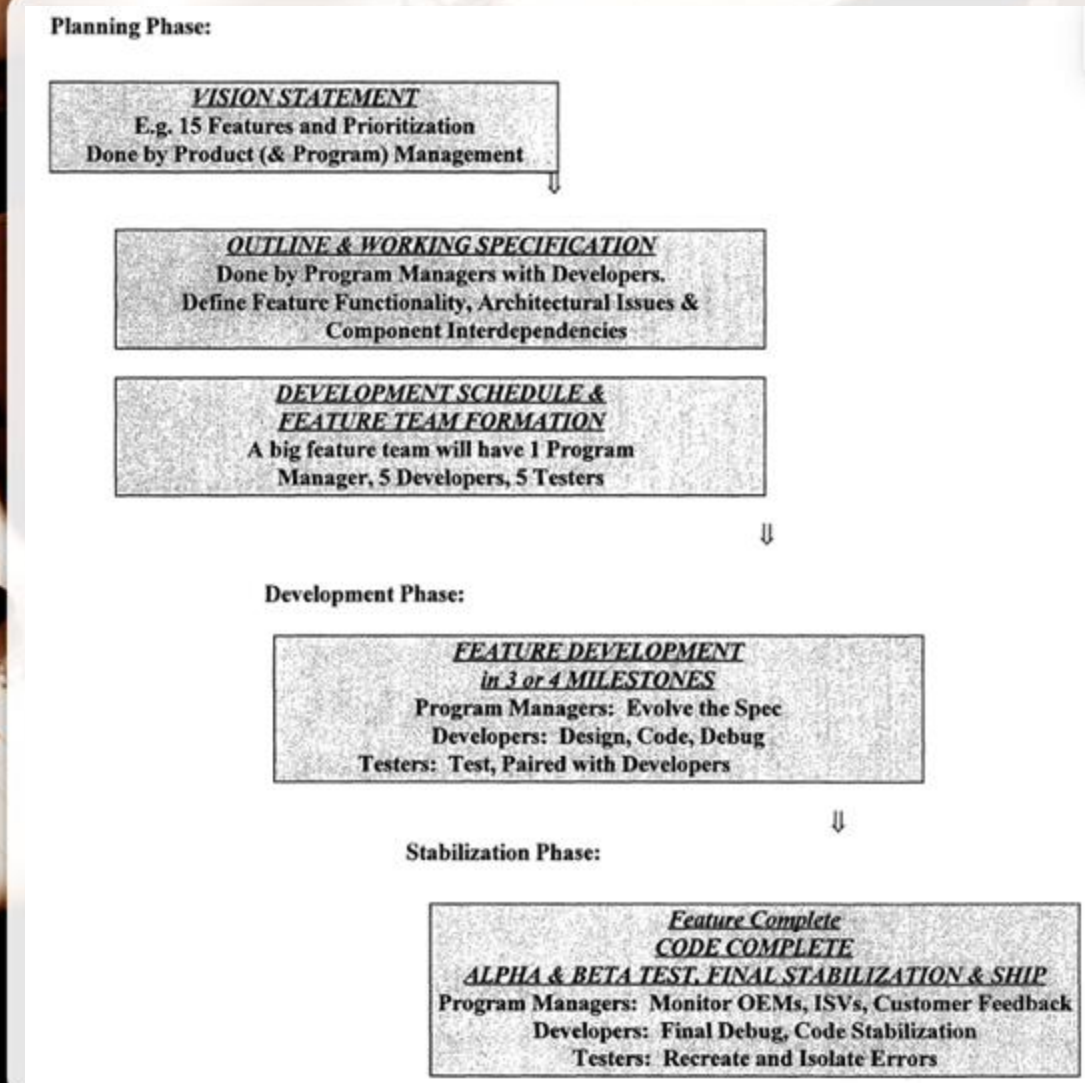
- Previous release provides feedback for the next[2].
- Feedbacks come from customer's comment on demonstrations of the software[2].
- This model results in more rapid delivery and deployment of software.
- Organizations such as Microsoft and HP applied incremental model to their software projects. HP and Microsoft applied ID with good results[2].
- **Review yourself (2)** : Incremental model is said to be an improvement to Waterfall. Describe the improvement.

ID at Microsoft

- A substantial planning (3-12 mths) is followed by stabilization (debugging and integration)
- Increments are sequential at the top level.
- At lower level, increments are overlapped. Each developers code and integrate everyday with a buddy tester.
-

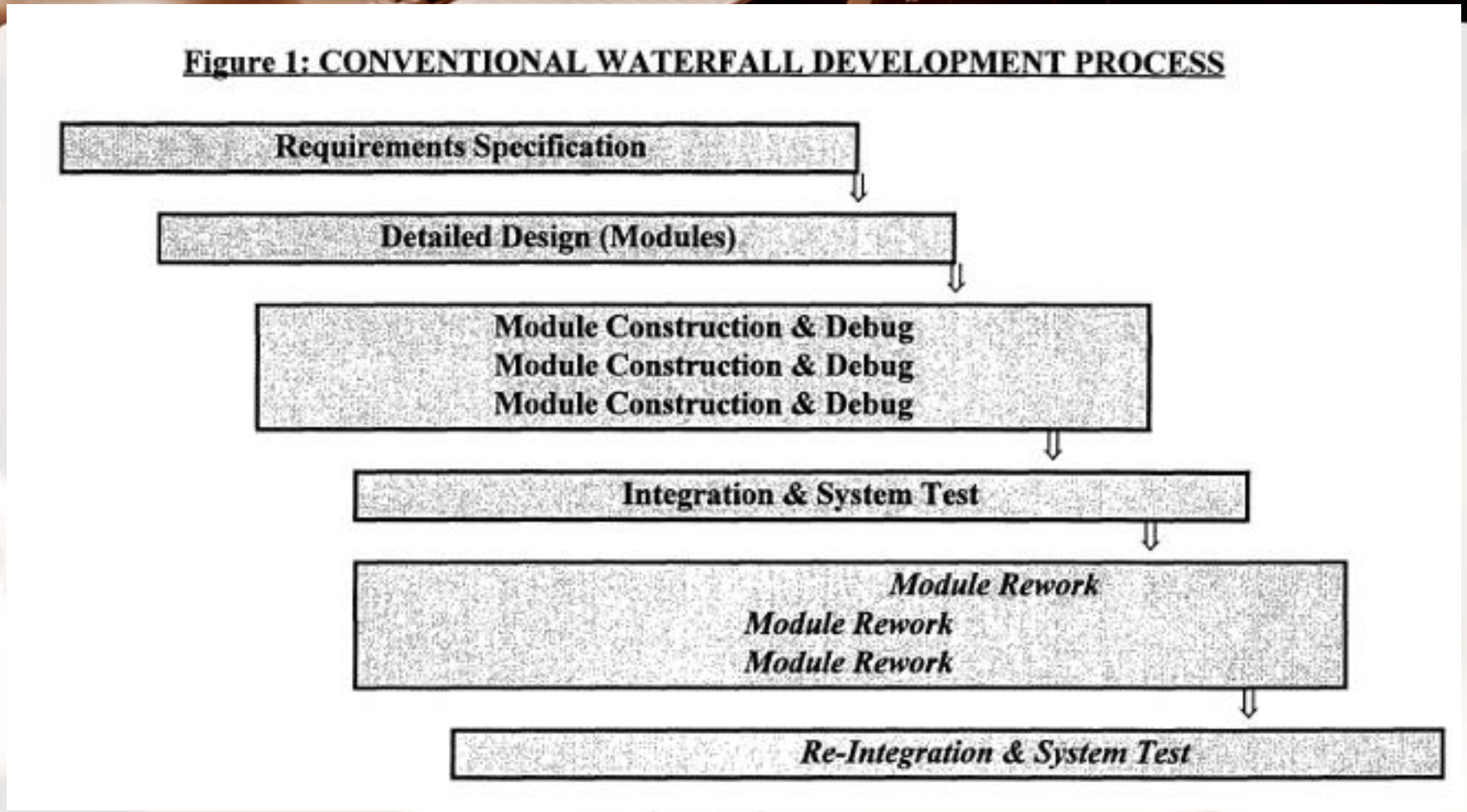


Microsoft's Synch & Stabilization[2]



- 12 or 24 months cycle

Conventional Waterfall Process[2]

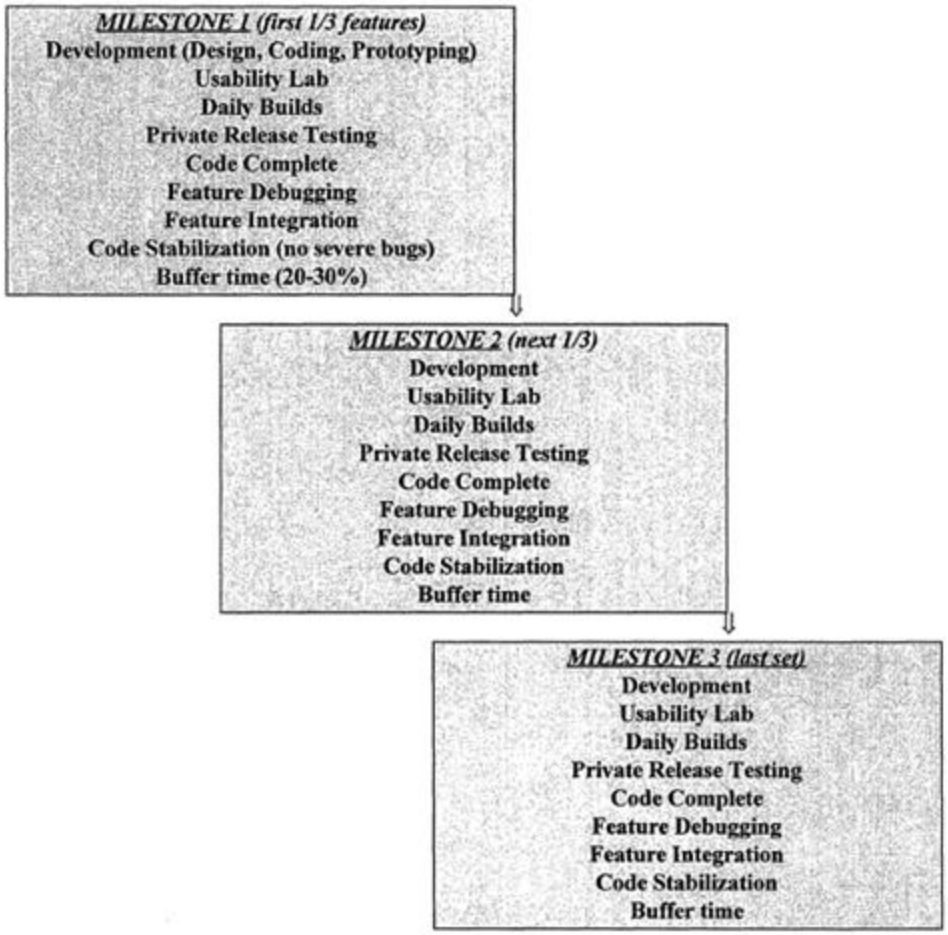


- A

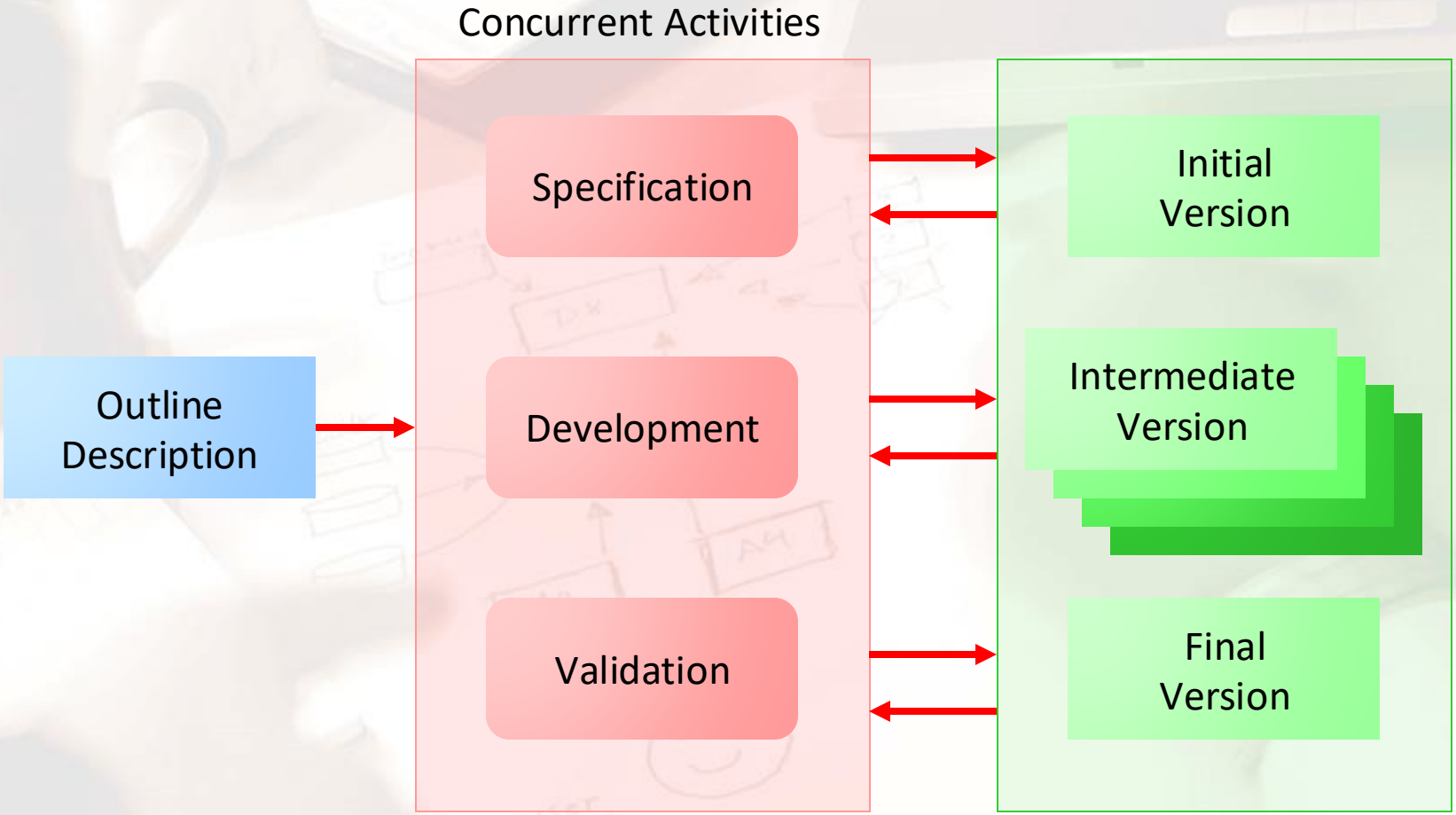
Synch & Stabilization's Milestone Breakdowns[2]

Figure 3: MICROSOFT'S "SYNCH-&-STABILIZE" MILESTONE BREAKDOWNS

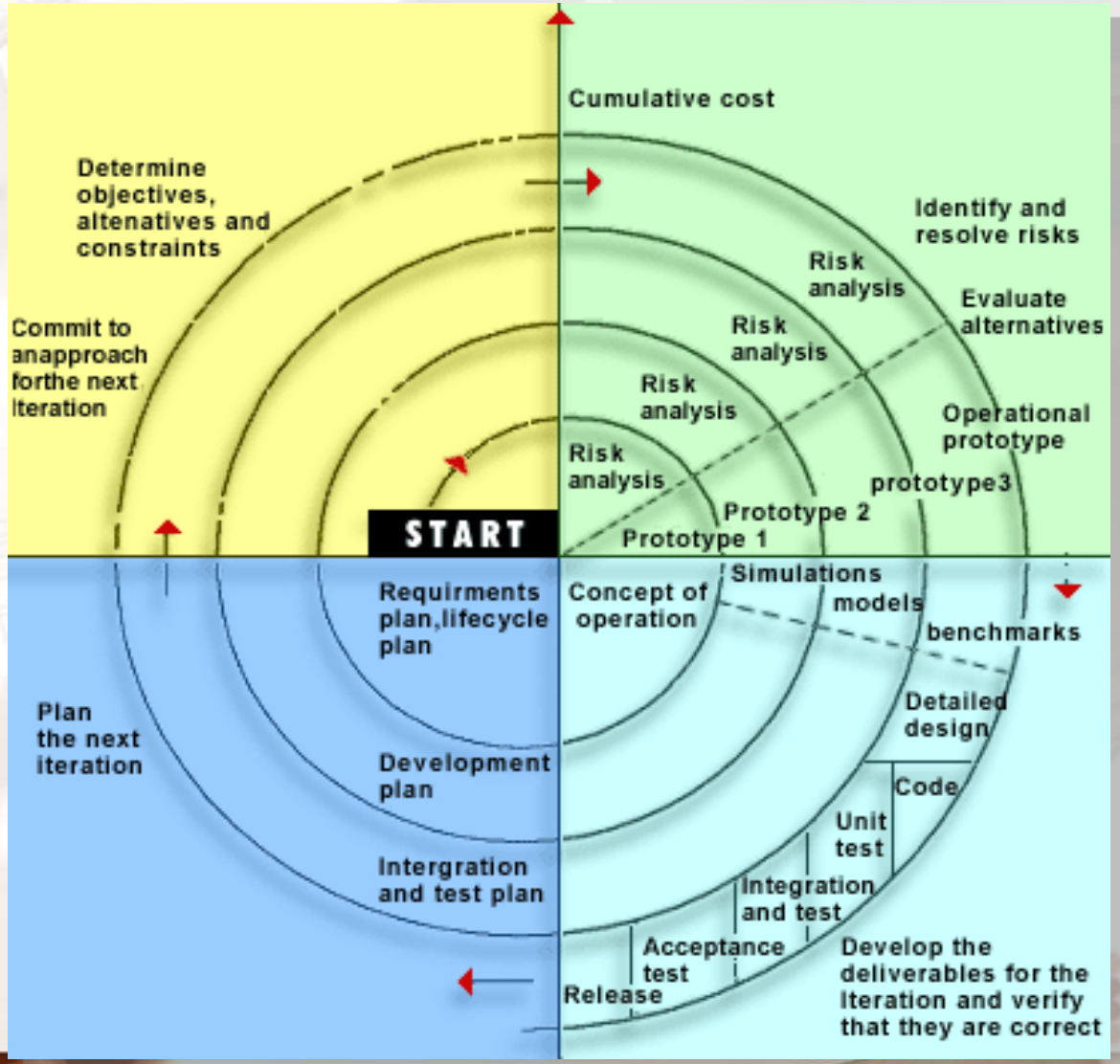
Time: Usually 2 to 4 months per Milestone



Evolutionary Model



Spiral Model

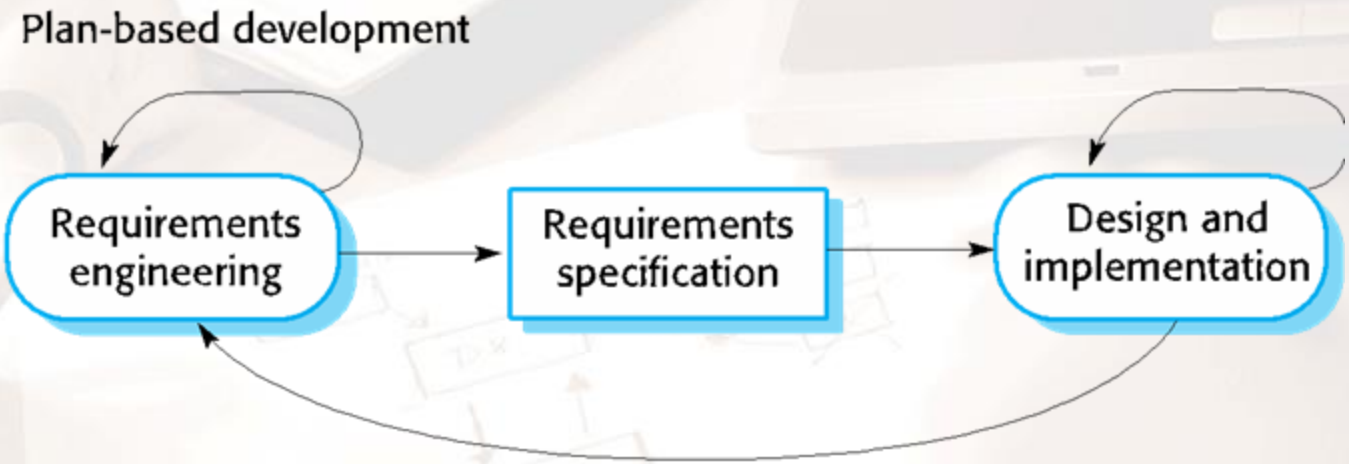


http://www.fusioninfotech.com/methodology.html

Agile Software Development

- "Rapid development and delivery is now often the most important requirement for software systems" – *Sommerville, Software Engineering*
  - Fast changing requirement
  - Unclear and unknown requirements
- Agile methods were emerged in the late 1990s to reduce the delivery time for software releases.

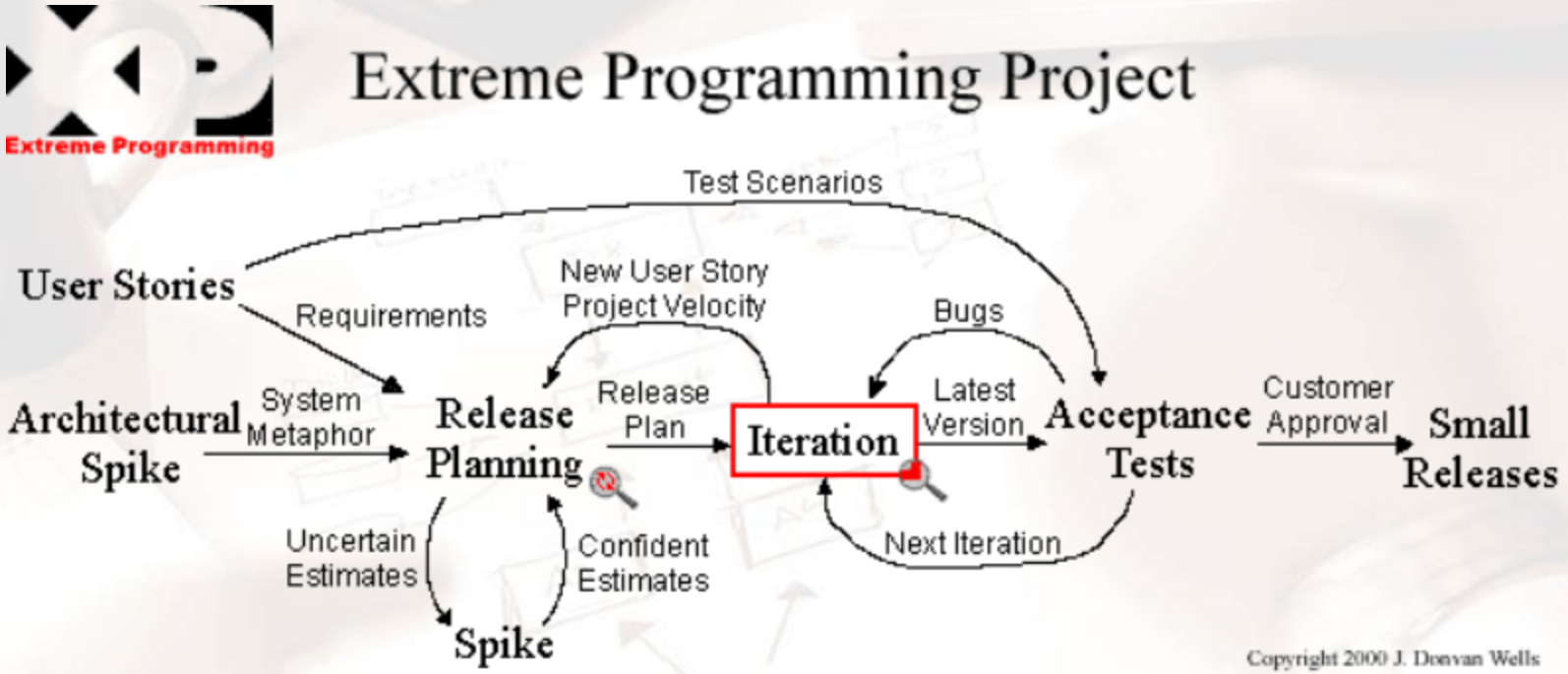
Plan-Driven & Agile



Agile Methodologies

- Extreme programming (XP)
- Scrum
- Feature-Driven Development (FDD)
- Agile Unified Process (AUP)

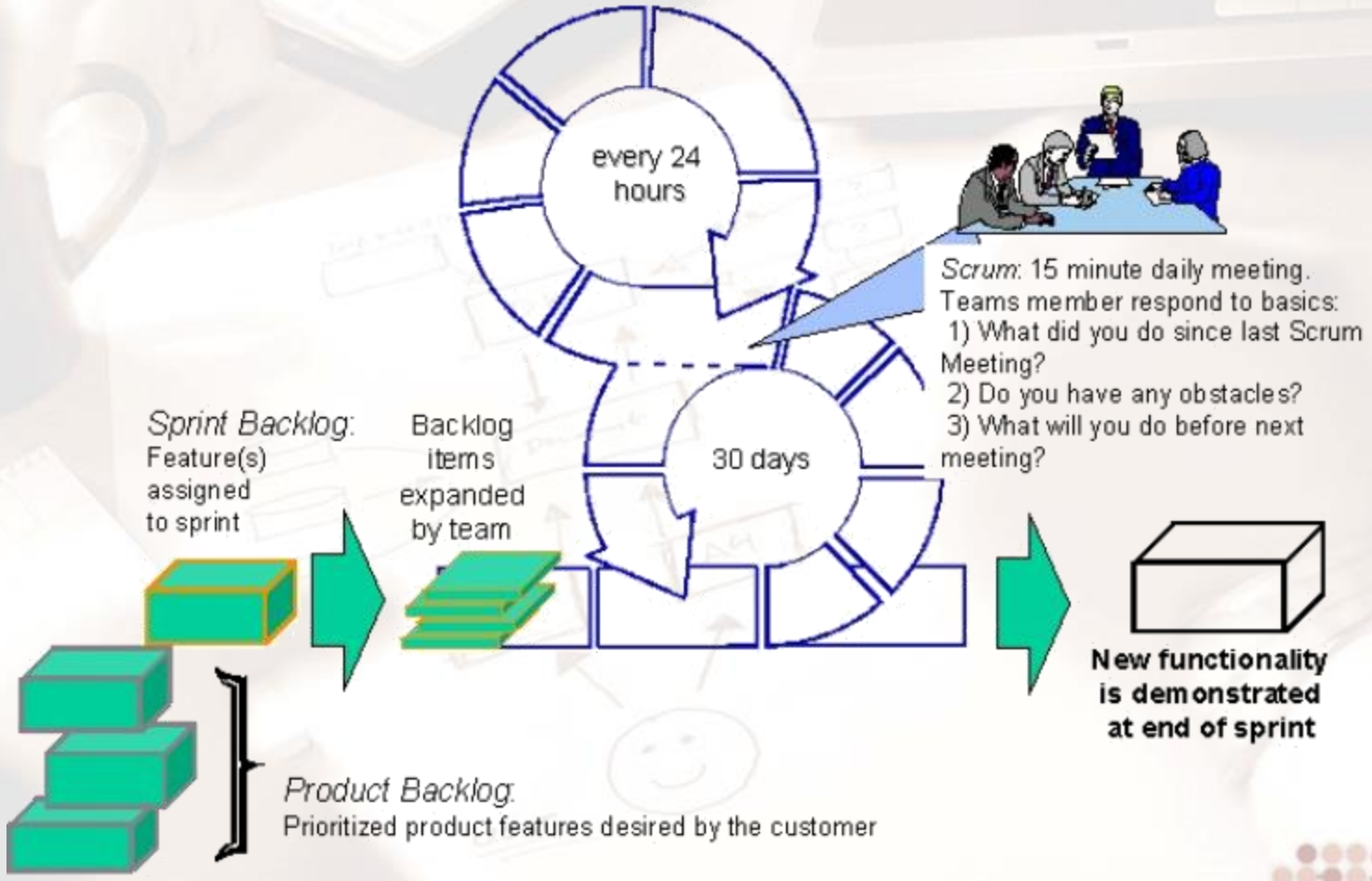
Extreme Programming



XP Issues

- Lowers the cost of changes by embracing new requirements, and perform most of the design activity incrementally and on the fly.
- Pair programming may be difficult for some practitioners.
- There is no up-front “detailed design”, which could result in more redesign effort in the long run.

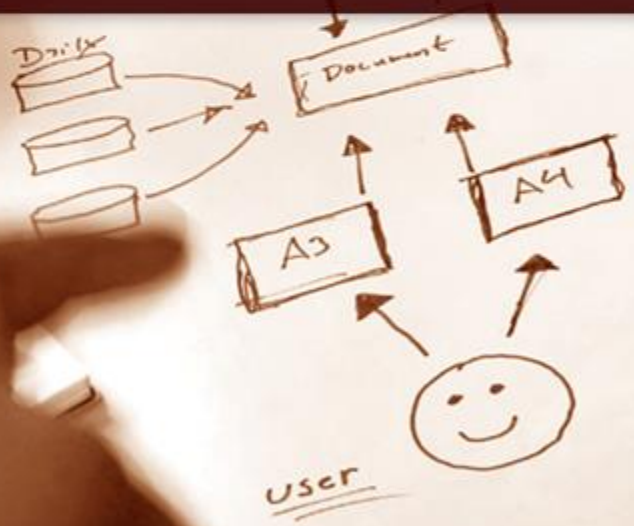
Scrum


every 24 hours
30 days
Scrum: 15 minute daily meeting.
Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

Scrum issues

- Improvement in team productivity, work prioritization, utilization of sprint, daily measured progress and communications.
- However, Scrum relies on the master to facilitate the team. Internal power struggles may paralyze the team.

Software Development



Software Process Activities

Definition Phase	Development Phase	Support Phase

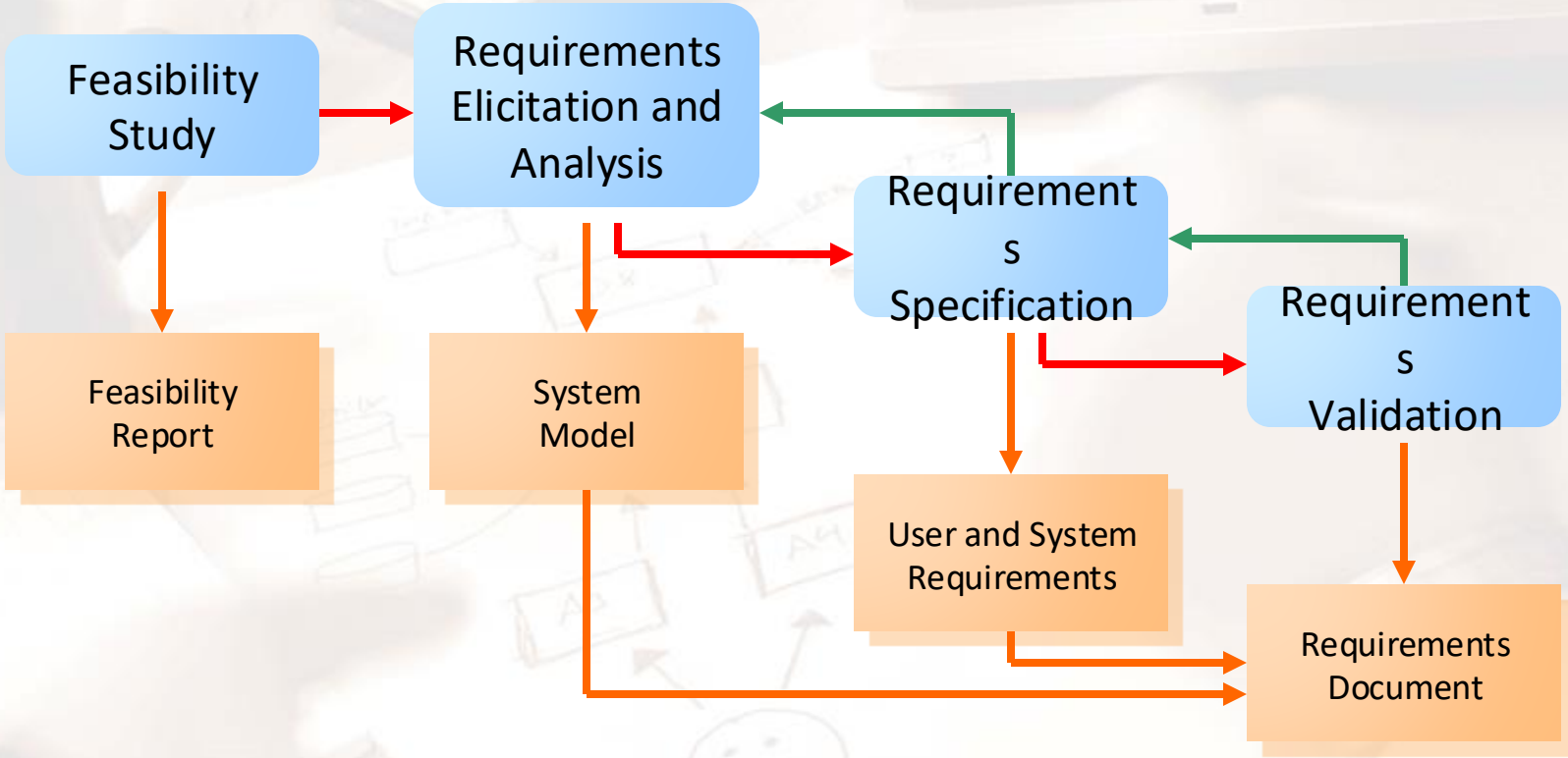
Definition Phase

Software Process Activities

Definition Phase	Development Phase	Support Phase

- Requirements definition
- Negotiation
- Functional specification
- System design

The Requirements Engineering Process



Requirement Definition (*IEEE*)

1. Introduction
2. Overall description
3. External interface requirements
4. System features
5. Other nonfunctional requirements
6. Other requirements

Requirement Definition (*Contd.*)

1. Introduction
   - Purpose
   - Document conventions
   - Intended audience
   - Additional information
   - Contact information
   - References

Requirement Definition (*Contd.*)

2. Overall Description
   - Product perspective
   - Product functions
   - User classes and characteristics
   - Operating environment
   - User environment
   - Design/implementation constraints
   - Assumptions and dependencies

Requirement Definition (*Contd.*)

3. External Interface Requirements
   - User interfaces
   - Hardware interfaces
   - Software interfaces
   - Communication protocols and interfaces

Requirement Definition (*Contd.*)

4. System Features
   - System feature A
     - Description and priority
     - Action/result
     - Functional requirements
   - System feature B

Requirement Definition (*Contd.*)

5. Other Nonfunctional Requirements
   - Performance requirements
   - Safety requirements
   - Security requirements
   - Software quality attributes
   - Project documentation
   - User documentation

Requirement Definition (*Contd.*)

6. Other Requirements
   - Terminology
   - Glossary
   - Definitions list

Negotiation

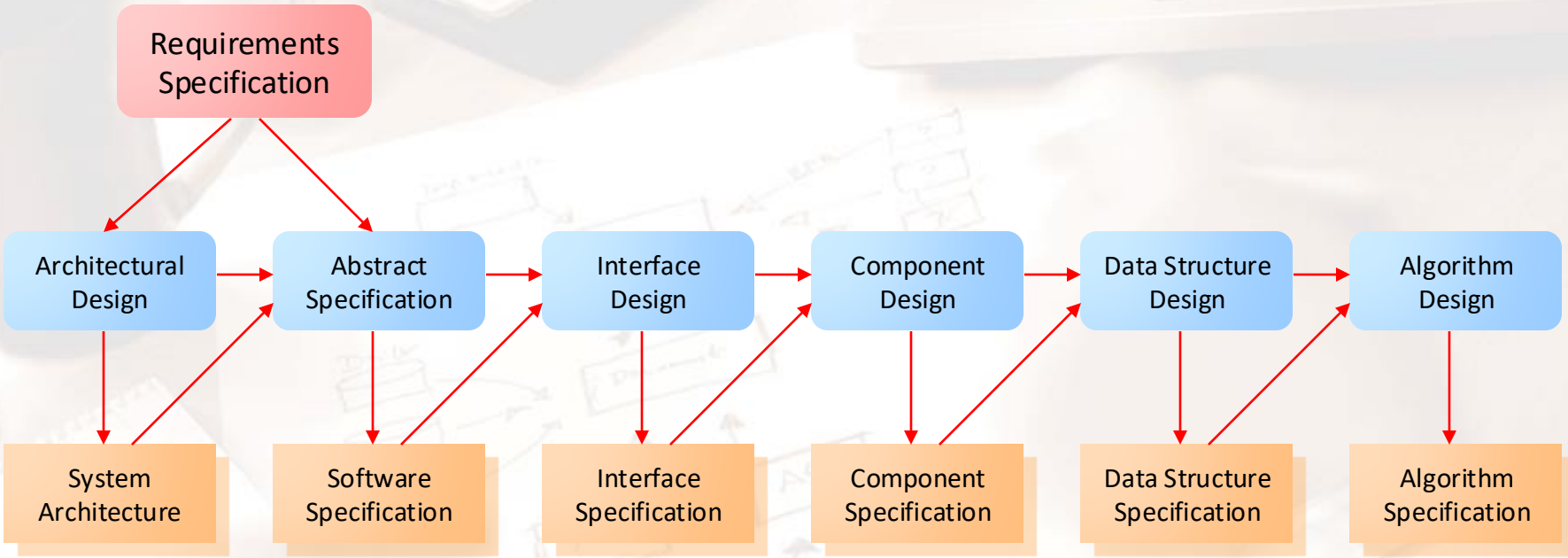
- Prepare, prepare, prepare
- Pay attention to timing
- Leave behind your ego
- Ramp up your listening skills
- If you don't ask, you don't get
- Anticipate compromise
- Offer and expect commitment
- Don't absorb their problems
- Stick to your principles
- Close with confirmation

Source: http://smallbusiness.yahoo.com/r-article-a-57774-m-1-sc-11-10_techniques_for_better_negotiation-i

Design Process Activities (*Sommerville*)

- Architectural design
- Abstract specification
- Interface design
- Component design
- Data structure design
- Algorithm design

Requirement Specification



Functional Specification (*Smith*)

- A functional specification is a technical document that specifies the functions that a system must perform
- Specifications may cover the following topics:
  - Objectives
  - Features
  - Users
  - Development process
  - Software model
  - Information flow
  - Standards and environments

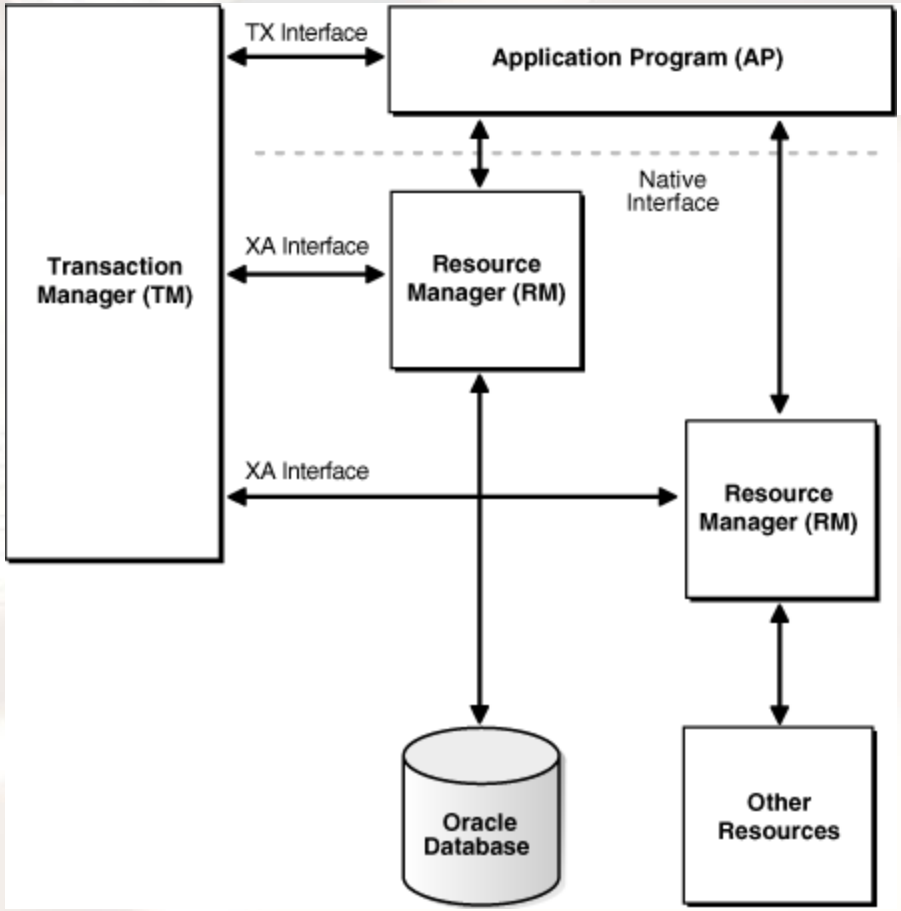
Source: Wikipedia.org

How to Write the Functional Specifications?

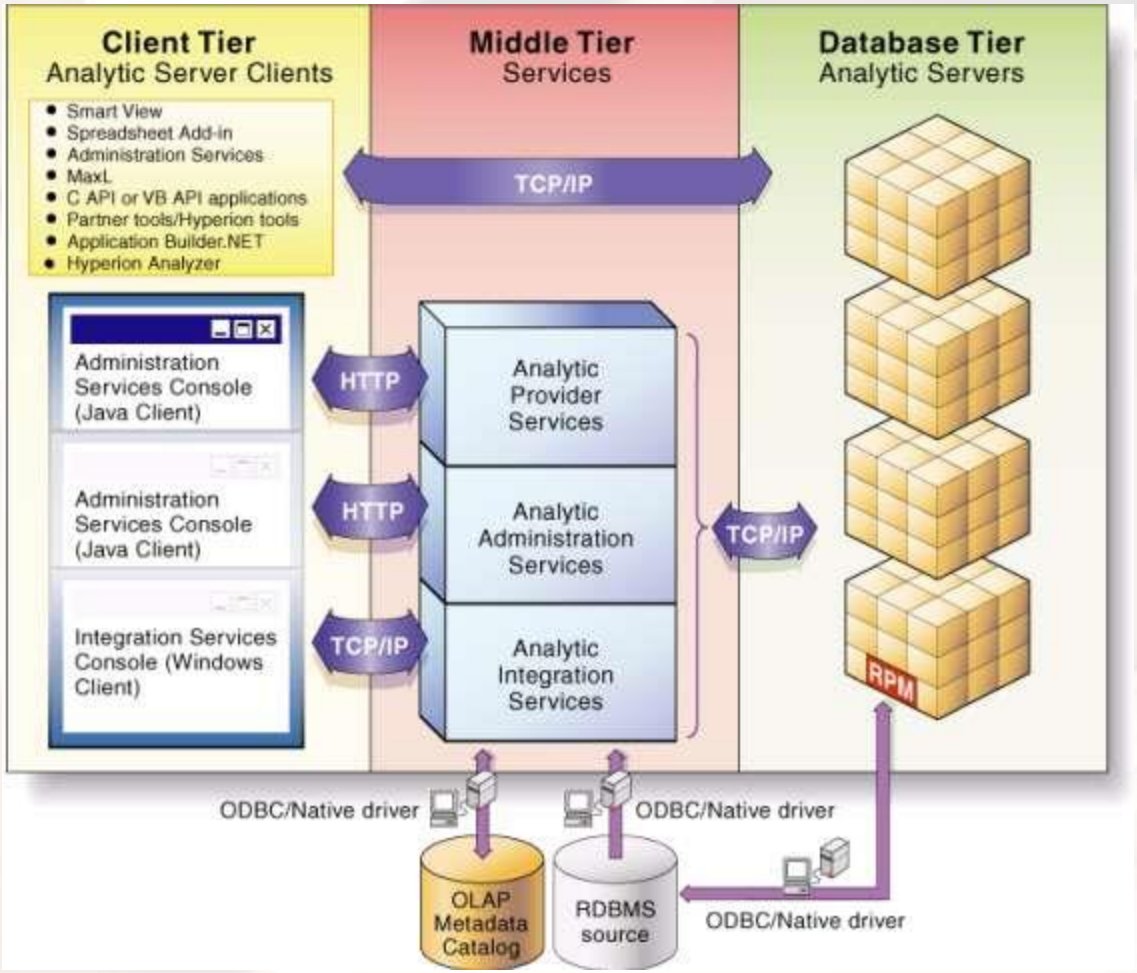
- Cover everything
- Use lots of tentative screenshots
- Write concisely, correctly and consistently
- Use the most comfortable tools and format
- Proofread and review

- **Example**

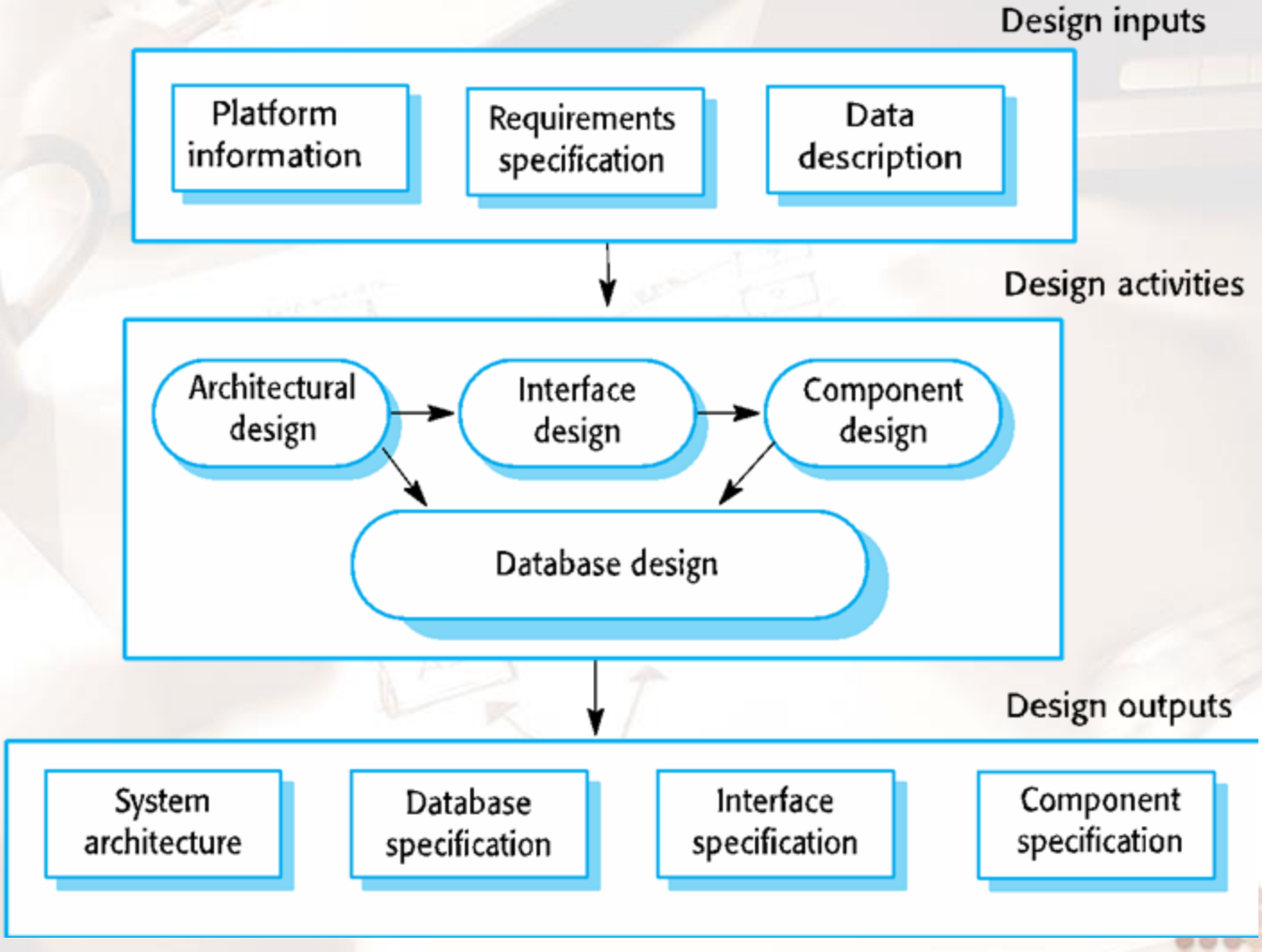
Software Model



Information Flow Diagram



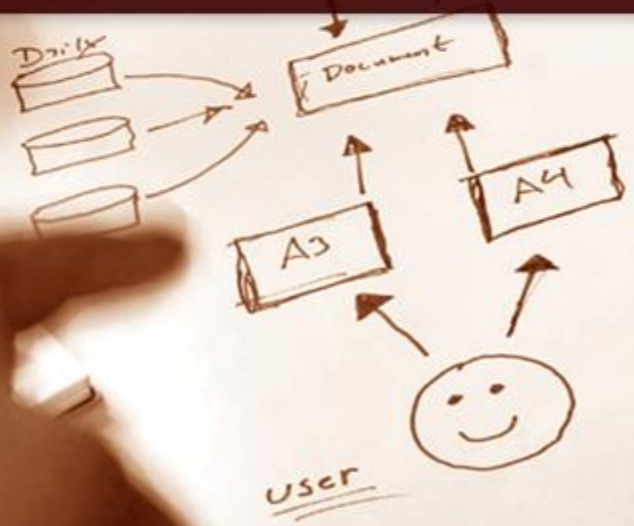
A General Design Process



Design Activities

- *Architectural design*
  - Identify the overall structure of the system, the principal components, their relationships and how they are distributed.
- *Database design*
  - The system data structures and how these are to be represented in a database.
- *Interface design*
  - The interfaces between system components.
- *Component selection and design*
  - Design how it will operate. Or select reusable components, ฯf available.

Development Phase



Software Process Activities

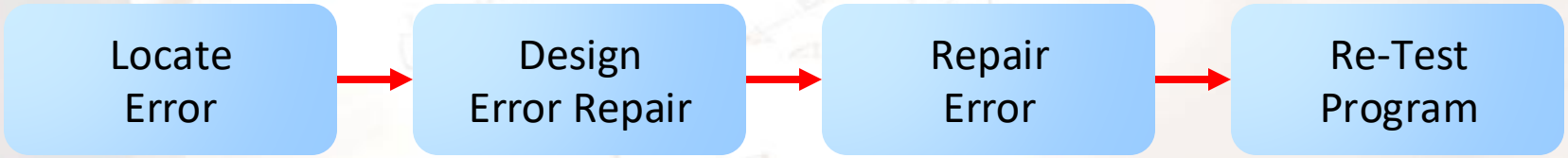
Definition Phase	**Development Phase**	Support Phase

- Sub-system development
- System integration
- System testing
- System validation and verification
- System installation

Programming and Debugging

- Translating a design into a program and removing errors from that program
- Programming is a personal activity - there is no generic programming process
- Programmers carry out some program testing to discover faults in the program and remove these faults in the debugging process

The Debugging Process



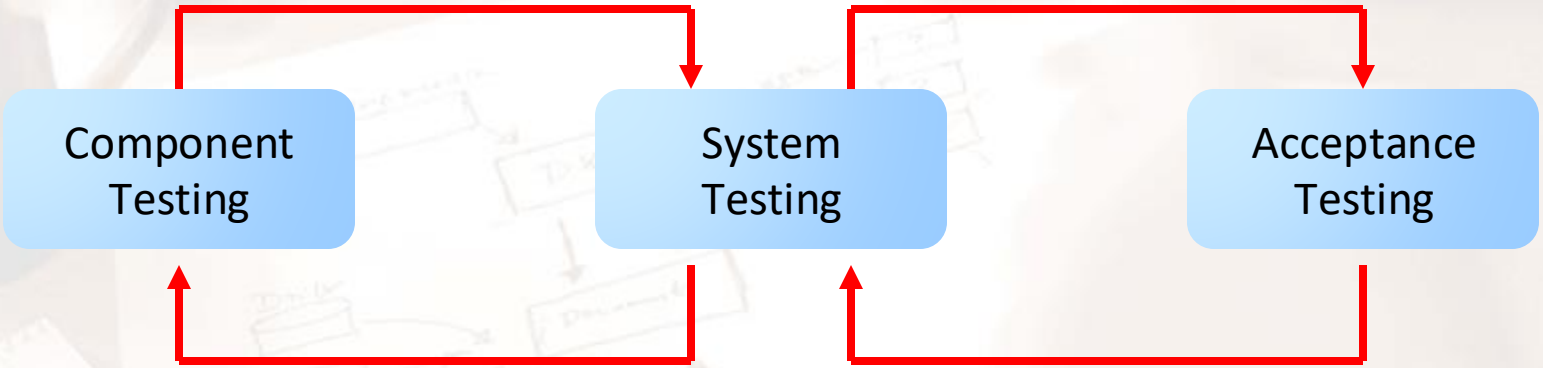
Software Testing

- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer
  - **Verification**: ensures the product is designed to deliver all functionality to the customer
  - **Validation**: ensures that functionality, as defined in requirements, is the intended behavior of the product
- Involves checking and review processes and system testing
- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system

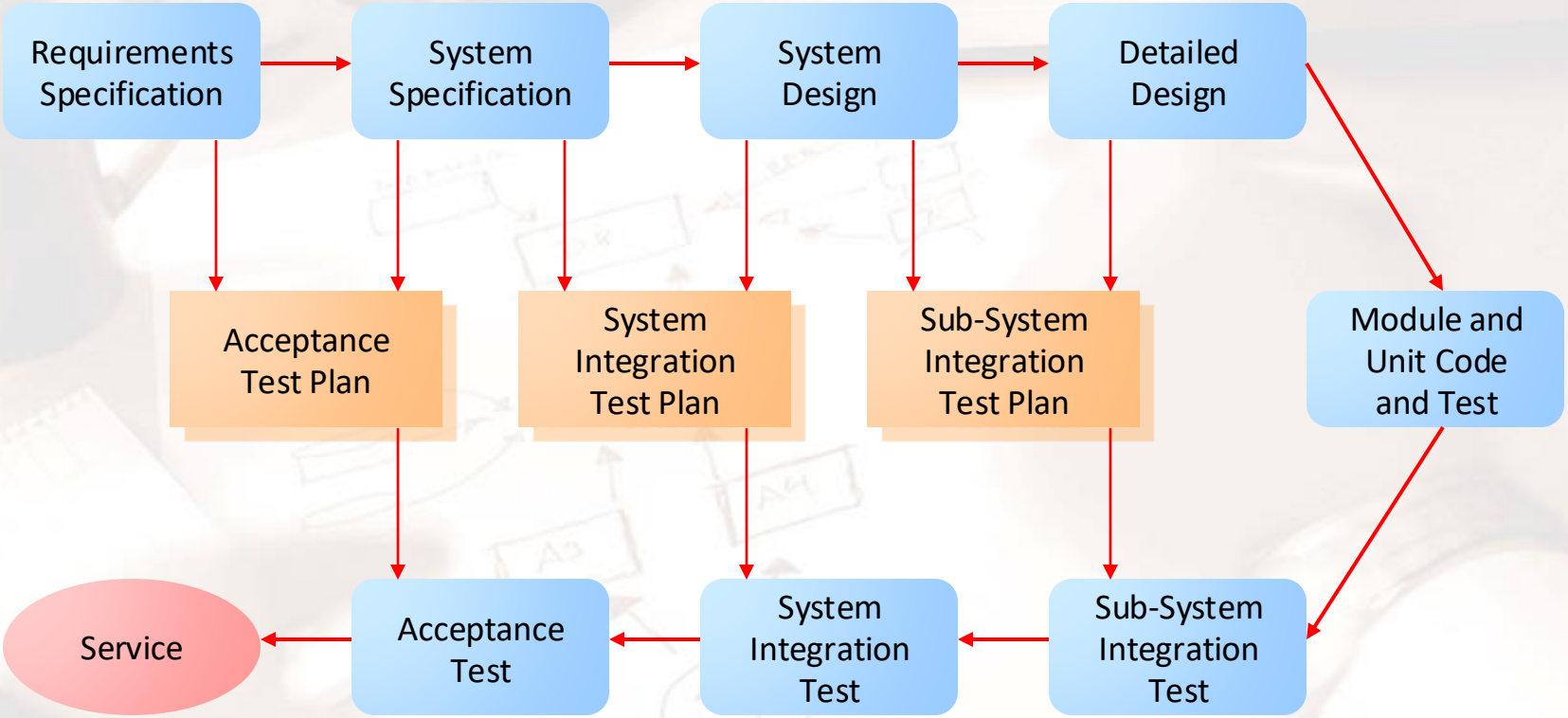
Testing Stages

- Component or unit testing
  - Individual components are tested independently
  - Components may be functions or objects or coherent groupings of these entities
- System testing
  - Testing of the system as a whole. Testing of emergent properties is particularly important.
- Acceptance testing
  - Testing with customer data to check that the system meets the customer's needs

The Testing Process



Testing Phases



Support Phase

Software Process Activities

Definition Phase	Development Phase	**Support Phase**

- System evolution
- System decommissioning
- Training
- Documentation
- Support

Supporting and Specification

- Should supporting activities be included in specification documents?

Software Evolution

- Software is inherently flexible and can change
- As requirements change through changing business circumstances, the software that supports the business must also evolve and change
- Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new

System Evolution