CSE 470 : Software Engineering

The Software Process

What is a Process?

- We can think of a series of activities as a **process**

- Any process has the following characteristics
  - It prescribes all of the major activities
  - It uses resources and produces intermediate and final products
  - It may include sub-processes and has entry and exit criteria
  - The activities are organized in a sequence
  - Constraints or control may apply to activities (budget control, availability of resources)

The Software Process

- **A structured set of activities required to develop a software system**
  - Specification
  - Analysis, design and implementation
  - Validation
  - Evolution
- **A software process model is an abstract representation of a process**
  - It presents a description of a process from some particular perspective

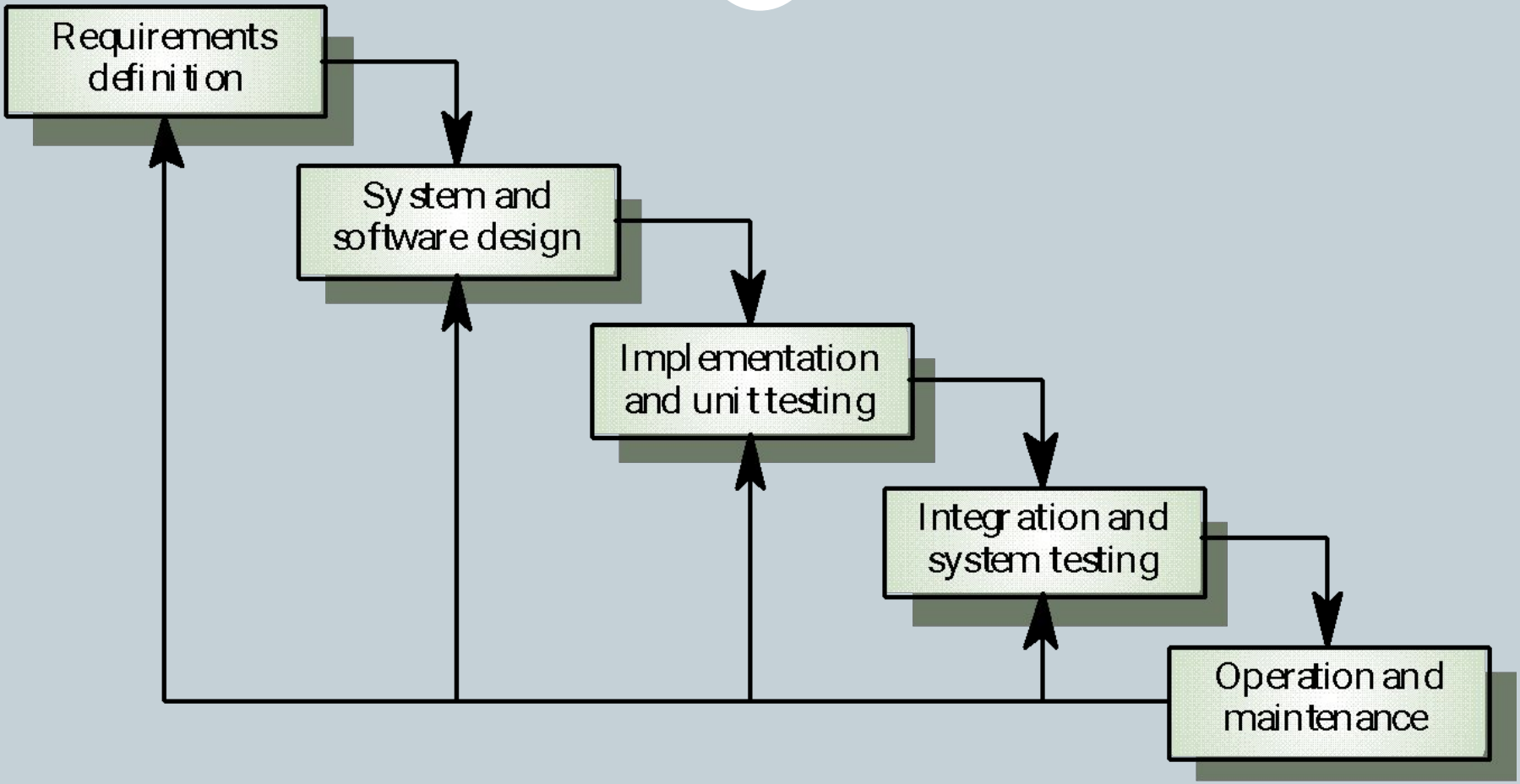
Software Development Approaches

- **Generic Software Process Models**
  1. **The waterfall model** (Linear Sequential Model)
     - Separate and distinct phases of specification and development
  2. **Evolutionary development**
     - Specification, development and testing are interleaved
- **Formal systems development (example - ASML)**
  - A mathematical system model is formally transformed to an implementation

ASML - Abstract State Machine Language

Yuri. Gurevich, Microsoft Research, 2001

Waterfall Model



Waterfall model phases

- **Requirements** analysis and definition
- System and software **design**
- **Implementation** and unit testing
- **Integration** and system testing
- **Operation** and maintenance

The drawback of the waterfall model is the difficulty of accommodating change after the process is underway

Waterfall model Requirement and Design

Artifacts produced in the requirements and Design phases

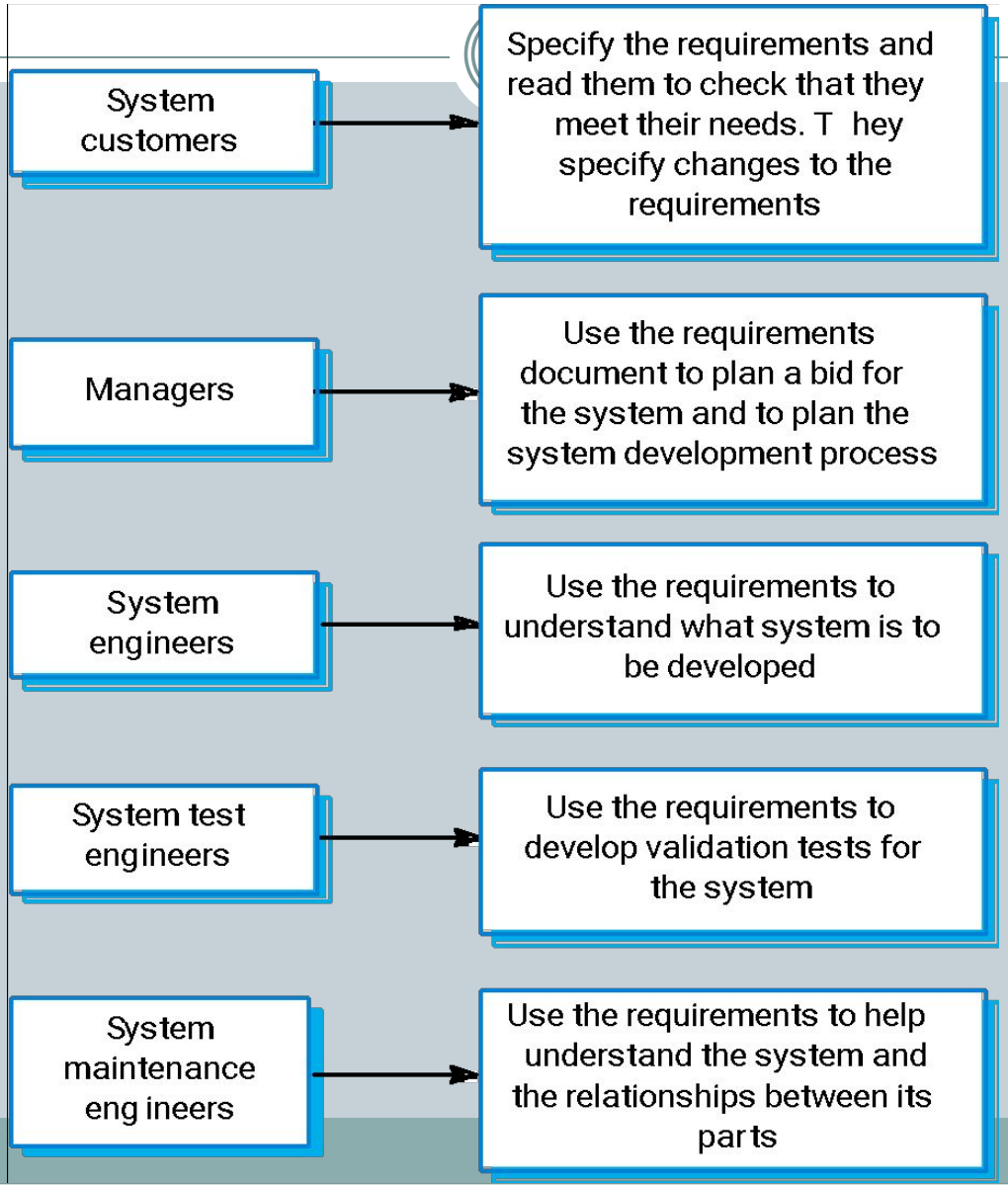
- **SRS -S**oftware **R**equirements **s**pecification document
- SRS might include:
  - User usage stories (scenarios) – **Use cases**.
  - Static analysis – **class diagrams**.
  - Behavioural analysis – **sequence diagrams**, **state charts**.

The specification and design activities are heavily time consuming.

The requirements document

- The requirements document is the official statement of what is required of the system developers.
- Should include both a definition of user requirements and a specification of the system requirements.
- **It is NOT a design document**. As far as possible, it should set of WHAT the system should do rather than HOW it should do it

Users of a requirements document



The Waterfall process characterization:

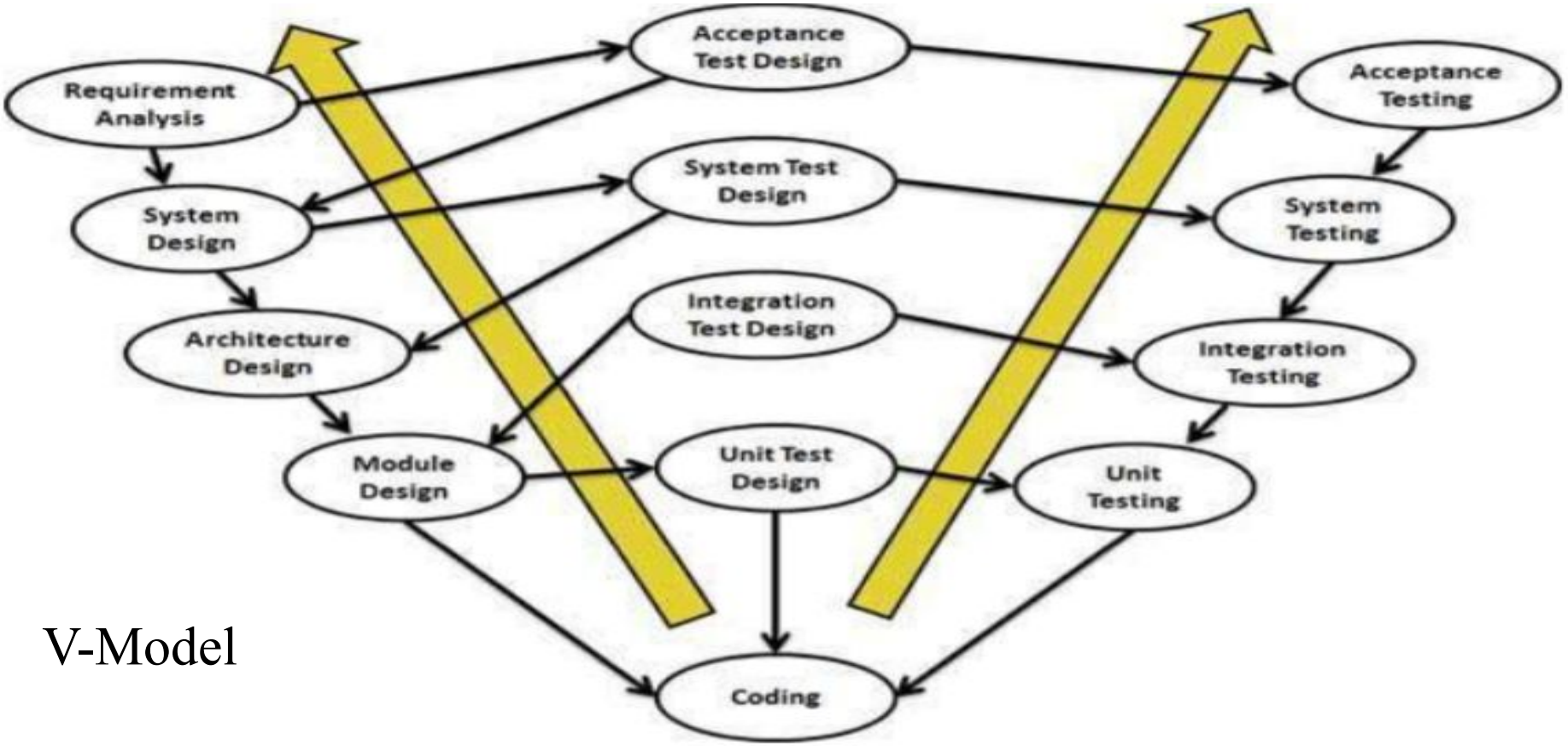
- Figure out what
- Figure out how
- Then do it
- Verify was done right
- Hand product to customer
- Perfect requirement definition?

Waterfall model problems

- **Inflexible partitioning** of the project into distinct stages
- Difficult to respond to changing customer requirements
- This model is only appropriate when the requirements are well-understood
- Waterfall model describes a staged development process
  - **Change during development is unlikely**
  - **Widely used in large systems: military and aerospace industries**

Why Not a Waterfall

- **No fabrication step**
  - Program code is another design level
  - Hence, no "commit" step – software can always be changed…!
- **No body of experience for design analysis (yet)**
  - Most analysis (testing) is done on program code
  - Hence, problems not detected until late in the process
- **Waterfall model takes a static view of requirements**
  - Ignore changing needs
  - Lack of user involvement once specification is written
- **Unrealistic separation of specification from the design**
- **Doesn't accommodate prototyping, reuse, etc.**

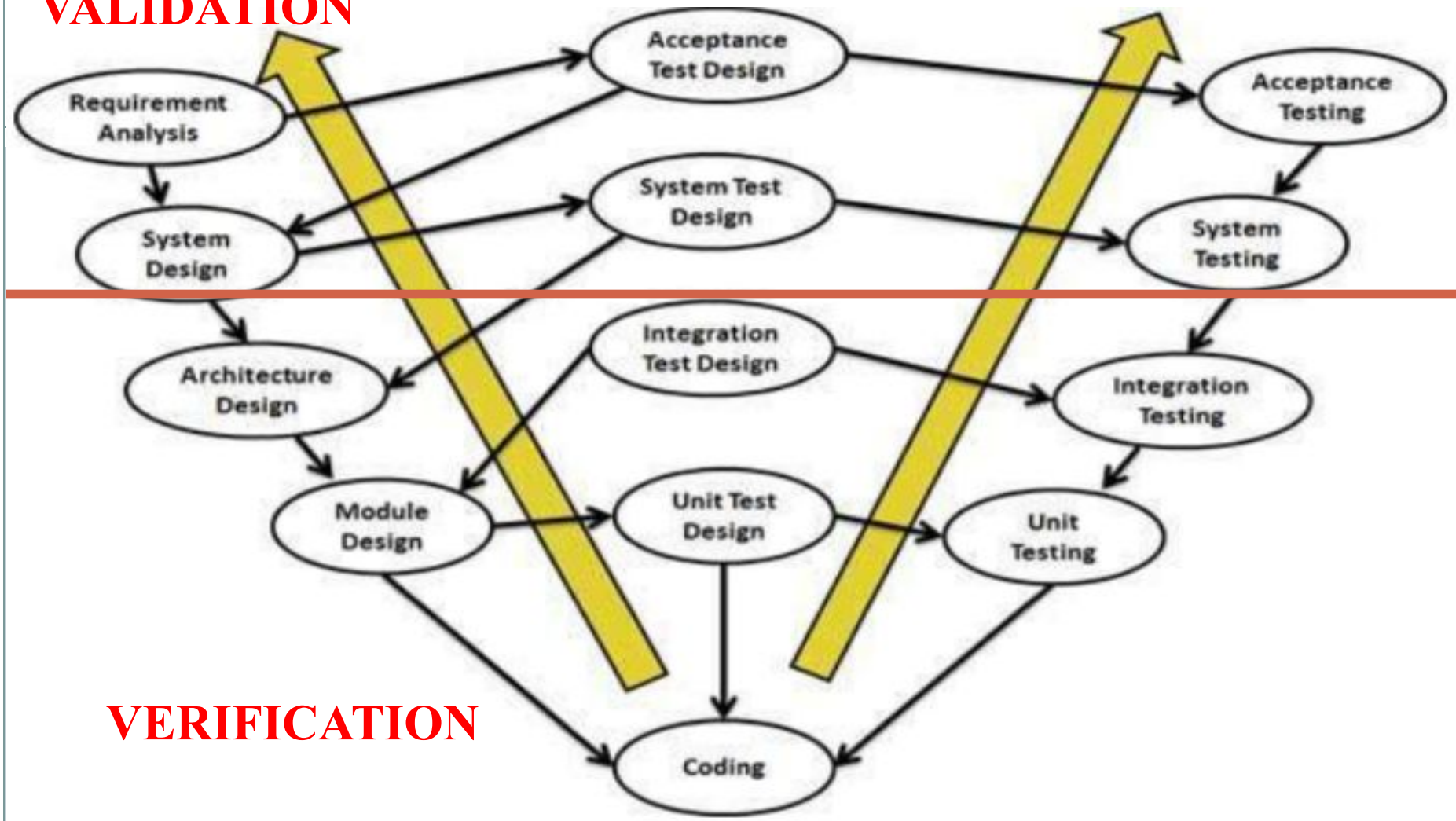


V-Model

- **The V - model is SDLC model where execution of processes happens in a sequential manner in V-shape. It is also known as Verification and Validation model.**
- **V - Model is an extension of the waterfall model and is based on association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle there is a directly associated testing phase. This is a highly disciplined model and next phase starts only after completion of the previous phase.**
- **Under V-Model, the corresponding testing phase of the development phase is planned in parallel. So there are Verification phases on one side of the .V. and Validation phases on the other side. Coding phase joins the two sides of the V-Model.**

V-Model (cont.)

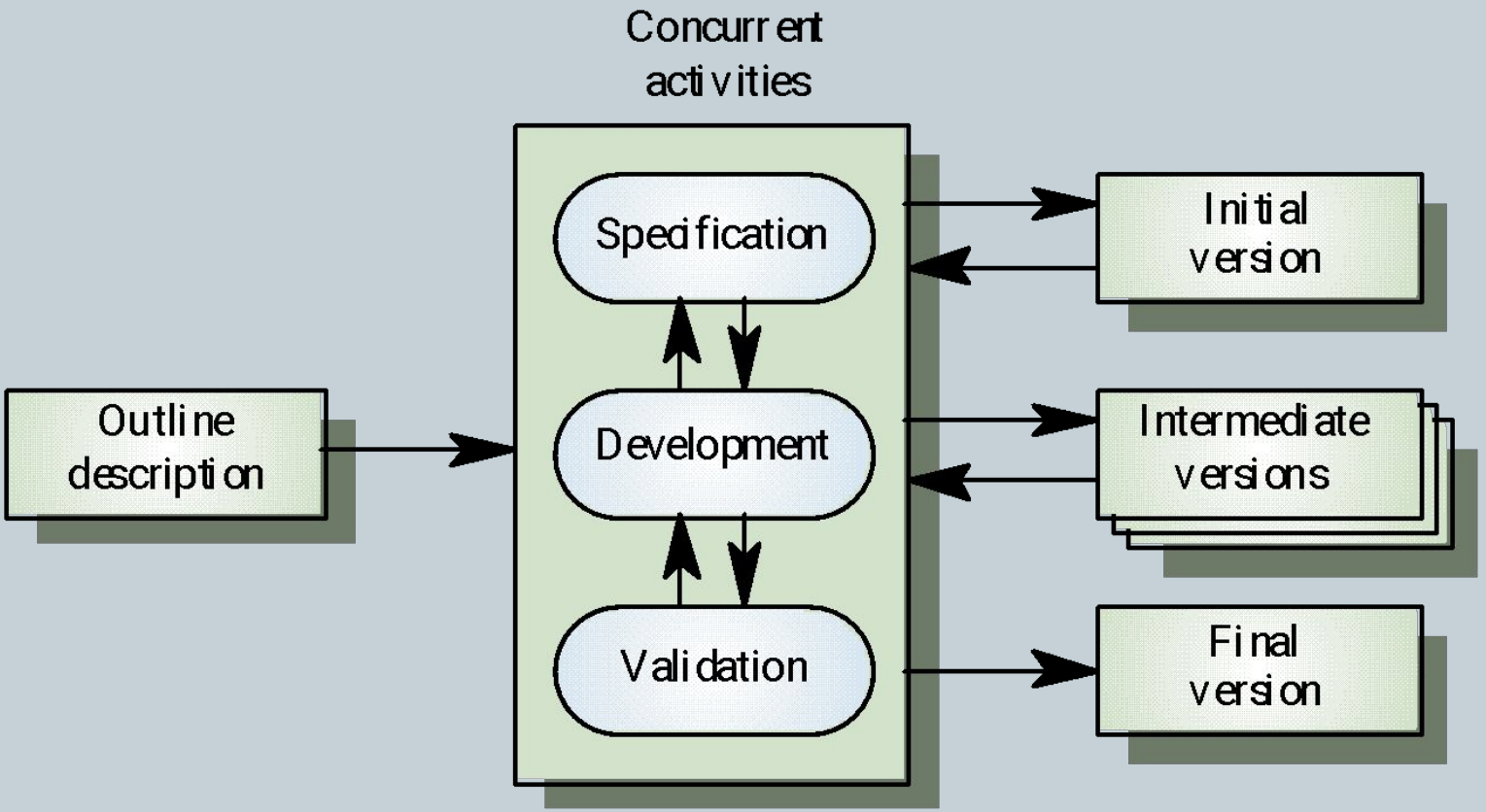
- **Verification Phases**
  - Do we build the product right?
- **Coding**
- **Validation Phases**
  - Do we build the right product?


VALIDATION
Acceptance Test Design
Requirement Analysis
Acceptance Testing
System Test Design
System Design
System Testing
Integration Test Design
Architecture Design
Integration Testing
Module Design
Unit Test Design
Unit Testing
VERIFICATION
Coding

V-Model (cont.)

Pros	Cons
• This is a highly disciplined model and Phases are completed one at a time. • Works well for smaller projects where requirements are very well understood. • Simple and easy to understand and use. • Easy to manage due to the rigidity of the model . each phase has specific deliverables and a review process.	• High risk and uncertainty. • Not a good model for complex and object-oriented projects. • Poor model for long and ongoing projects. • Not suitable for the projects where requirements are at a moderate to high risk of changing. • Once an application is in the testing stage, it is difficult to go back and change a functionality • No working software is produced until late during the life cycle.

Evolutionary development



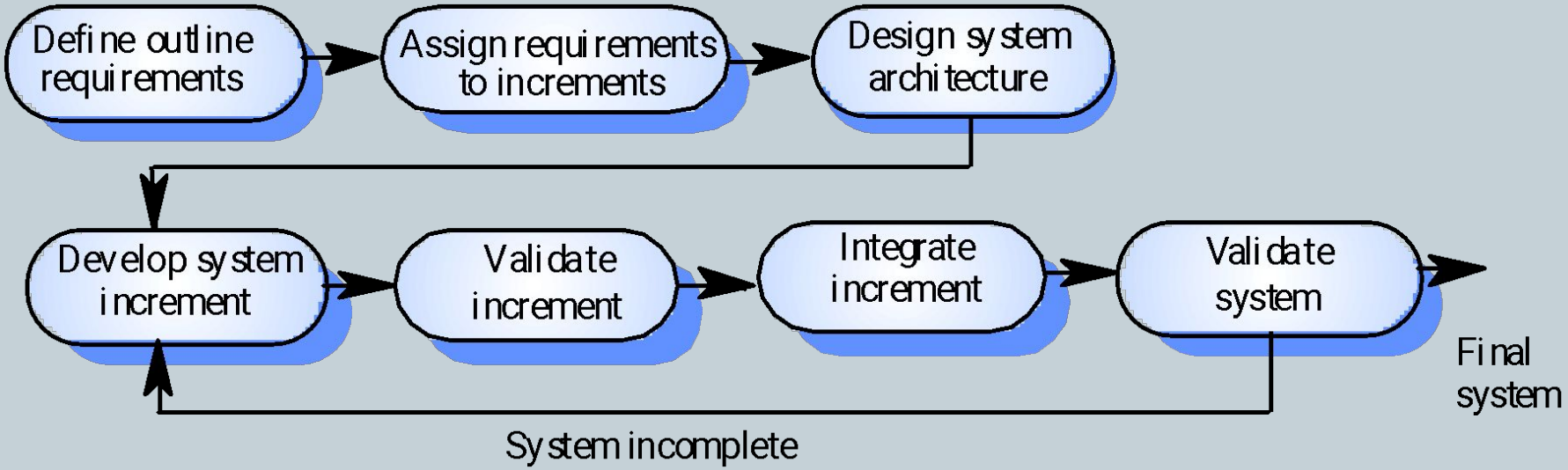
Characteristics of Evolutionary Development

- Modern development processes take ***evolution*** as fundamental, and try to provide ways of managing, rather than ignoring, the risk.
- System requirements always ***evolve*** in the course of a project.
- Specification is ***evolved*** in conjunction with the software – No complete specification in the system development contract. Difficult for large customers.
- Two (related) process models:
  - **Incremental development**
  - **Spiral development**

Incremental Development

- Rather than deliver the system as a single delivery, **the development and delivery is broken down into increments** with each increment delivering part of the required functionality.
- **User requirements are prioritised** and **the highest priority requirements are included in early increments.**
- **Once the development of an increment is started, the requirements are frozen** though requirements for later increments can continue to evolve.

Incremental Development – Version I



Incremental Development –Advantages

- **Customer value** can be delivered with each increment so system functionality is available earlier.
- **Early increments** act as a prototype to help elicit requirements for later increments.
- Lower risk of overall project failure.
- **The highest priority system services** tend to receive the most testing.

Incremental Development – Problems

- Lack of process visibility.
- Systems are often poorly structured.

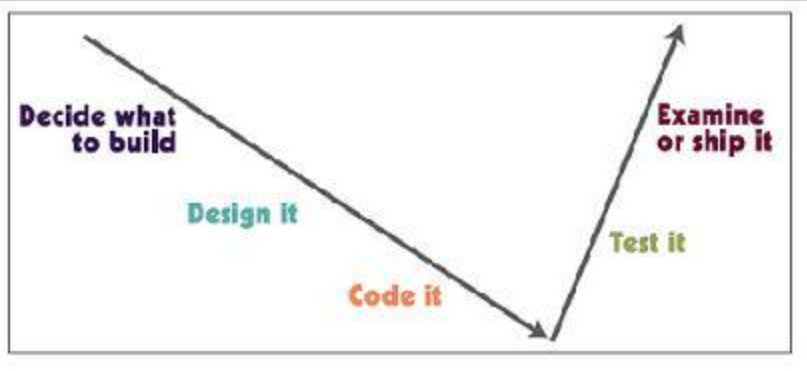
- **Applicability claims in the literature:**
  - For small or medium-size interactive systems.
  - For parts of large systems (e.g. the user interface).
  - Hardware or man power resources are limited.

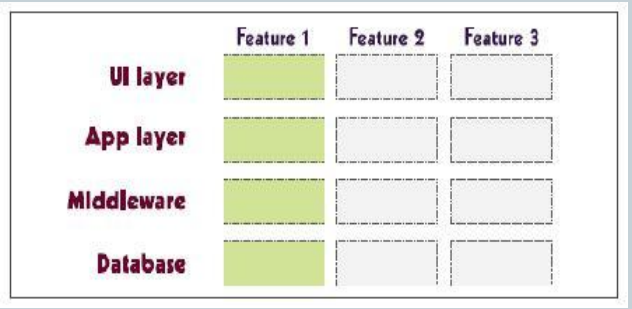
Incremental means adding, iterative means reworking (by Alistair Cockburn)

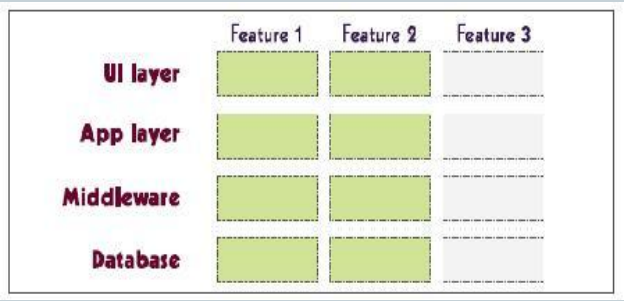
- **Incremental development** is a staging and scheduling strategy in which the various parts of the system are developed at different times or rates and integrated as they are completed. The alternative is to develop the entire system with a big bang integration at the end.
- **Iterative development** is a rework scheduling strategy in which time is set aside to revise and improve parts of the system. The alternative development is to get it right the first time (or at least declare that it is right!).

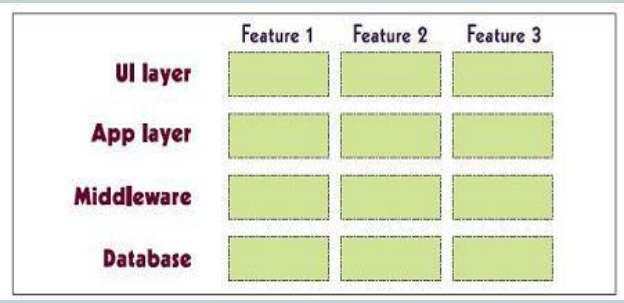
Iterate	Increment
fundamentally means *“change”.*	fundamentally means *“add”.* *onto*
repeating the process on the *same* section of work	repeating the process on a *new* section of work
repeat the process (, design, ,implement, evaluate)	repeat the process (, design, ,implement, evaluate)

Incremental Development



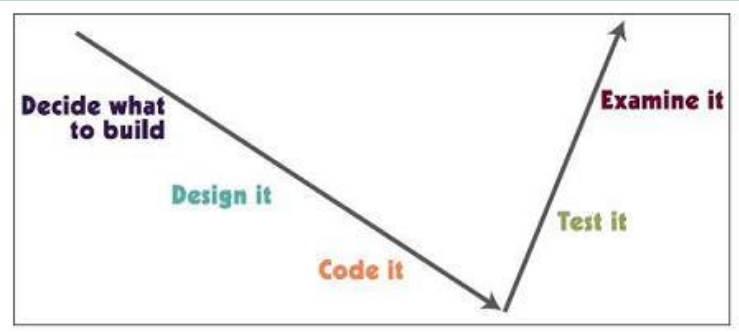




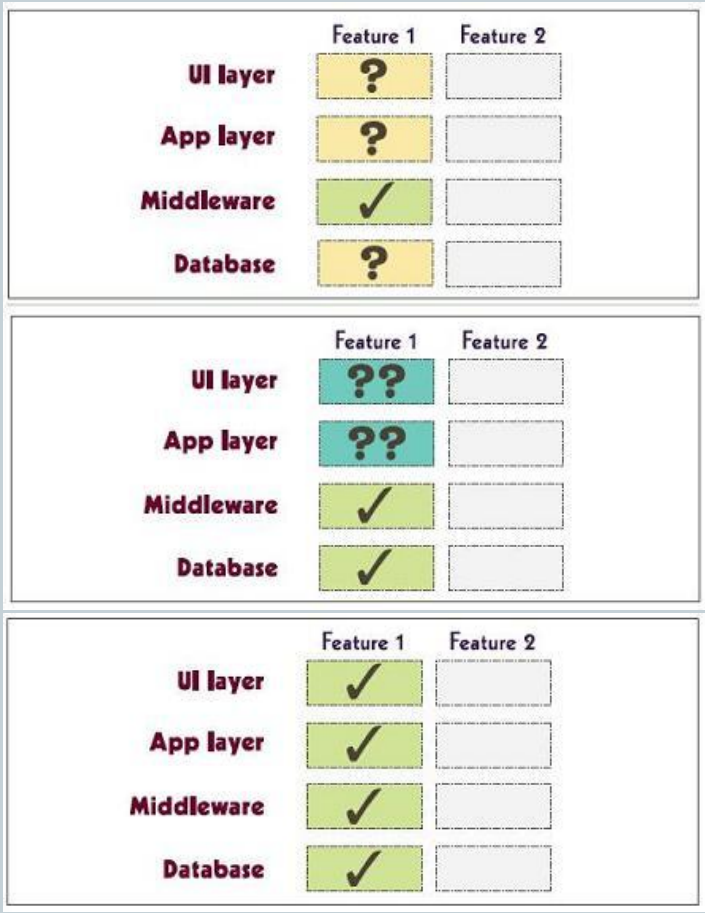


- **The first increment delivers one slice of functionality through the whole system.**
- **The second increment delivers another slice of functionality through the whole system.**
- **The third increment delivers the rest of the system**

Iterative Development



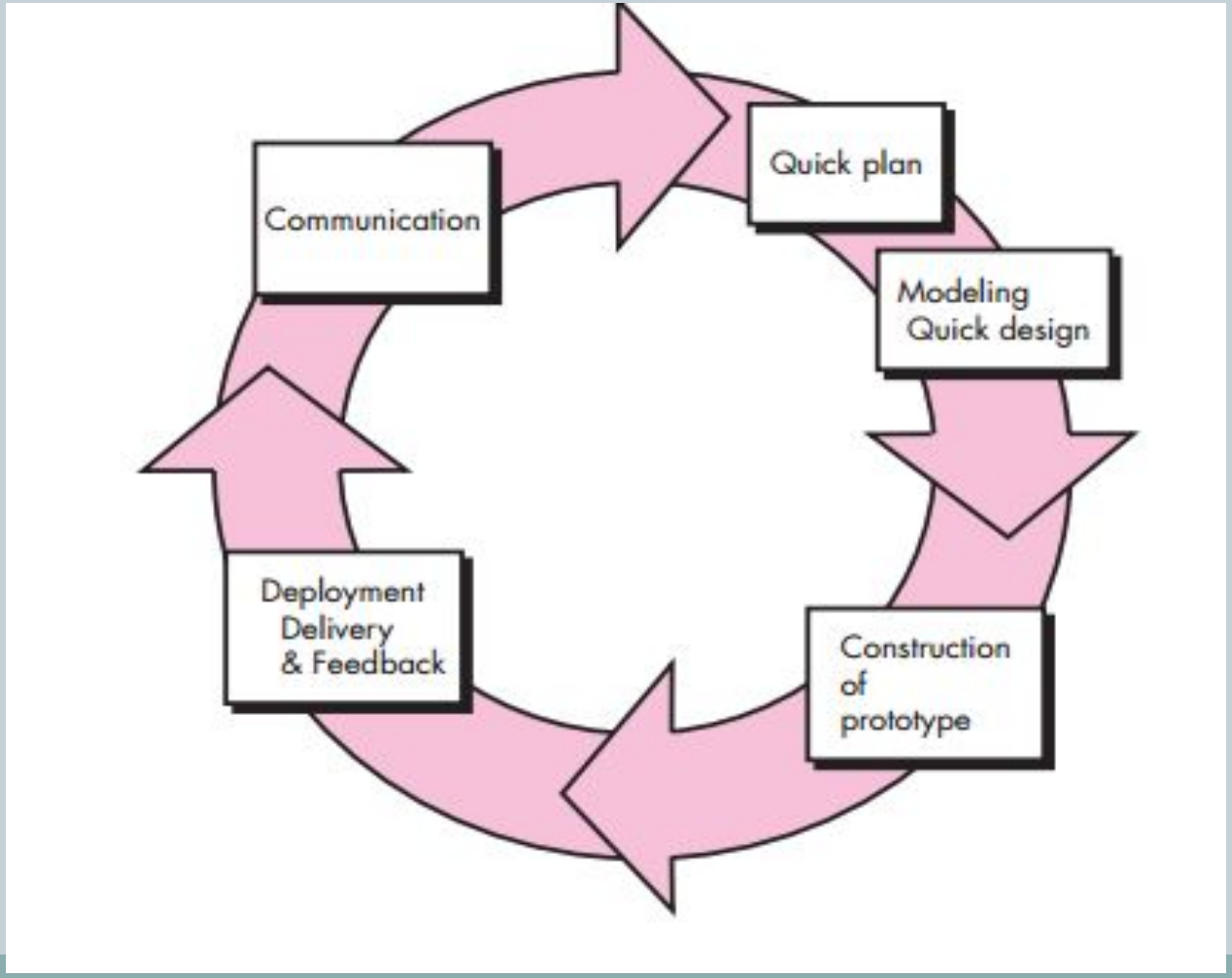
- **The first iteration delivers enough of feature 1 to evaluate what is correct and what needs revision.**
- **After the second iteration, some revised parts still need improvement.**
- **The third iteration produces the final and stable feature**



Iterative Development – Prototyping Model

- Sometimes user don't know the requirement clearly.
- May be confused about the efficiency of an algorithm, the adaptability of an operating system, or the form that human-machine interaction should take.
- The delivered product is a prototype initially (not a running product).
- Ideally, the prototype serves as a mechanism for identifying software requirements

Iterative Development – Prototyping Model



Iterative Development – Prototyping Model

- Disadvantages:
  - The stakeholders see the ready product, and don't know it has been created haphazardly by gluing several resources to deliver asap.
  - Using inefficient approaches seem normal, and long lasts in the system code.

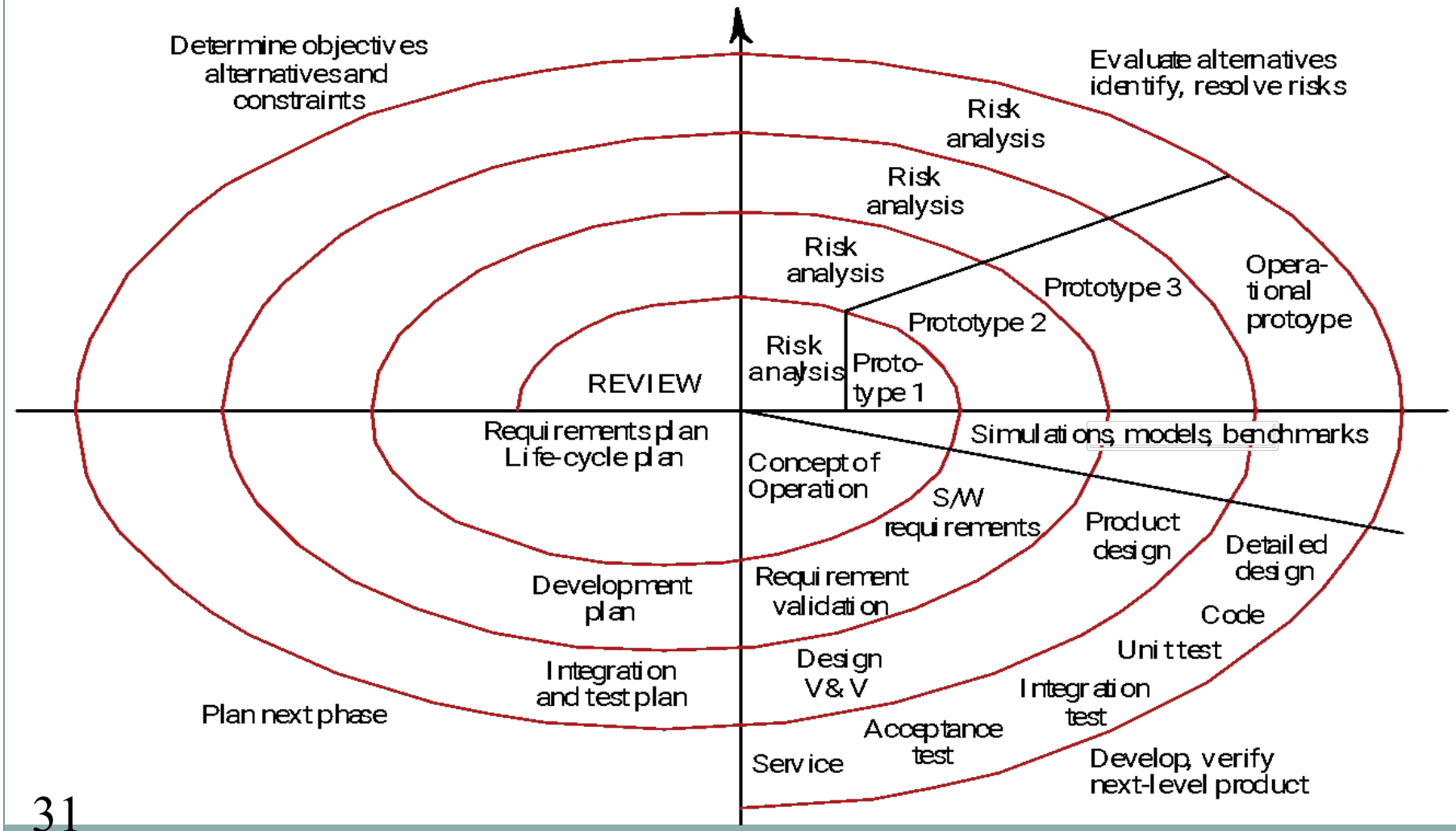
Spiral Model

- The **spiral model** is a **risk-driven** process model process model generator for software projects. Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as incremental process model generator for software projects. Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as incremental, waterfall process model generator for software projects. Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as incremental, waterfall, or evolutionary prototyping.

Spiral development

- Process is represented as a spiral rather than as a sequence of activities with backtracking.
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design – loops in the spiral are chosen depending on what is required.
- **Risks** are explicitly assessed and resolved throughout the process.

Spiral model of the software process



Spiral model sectors

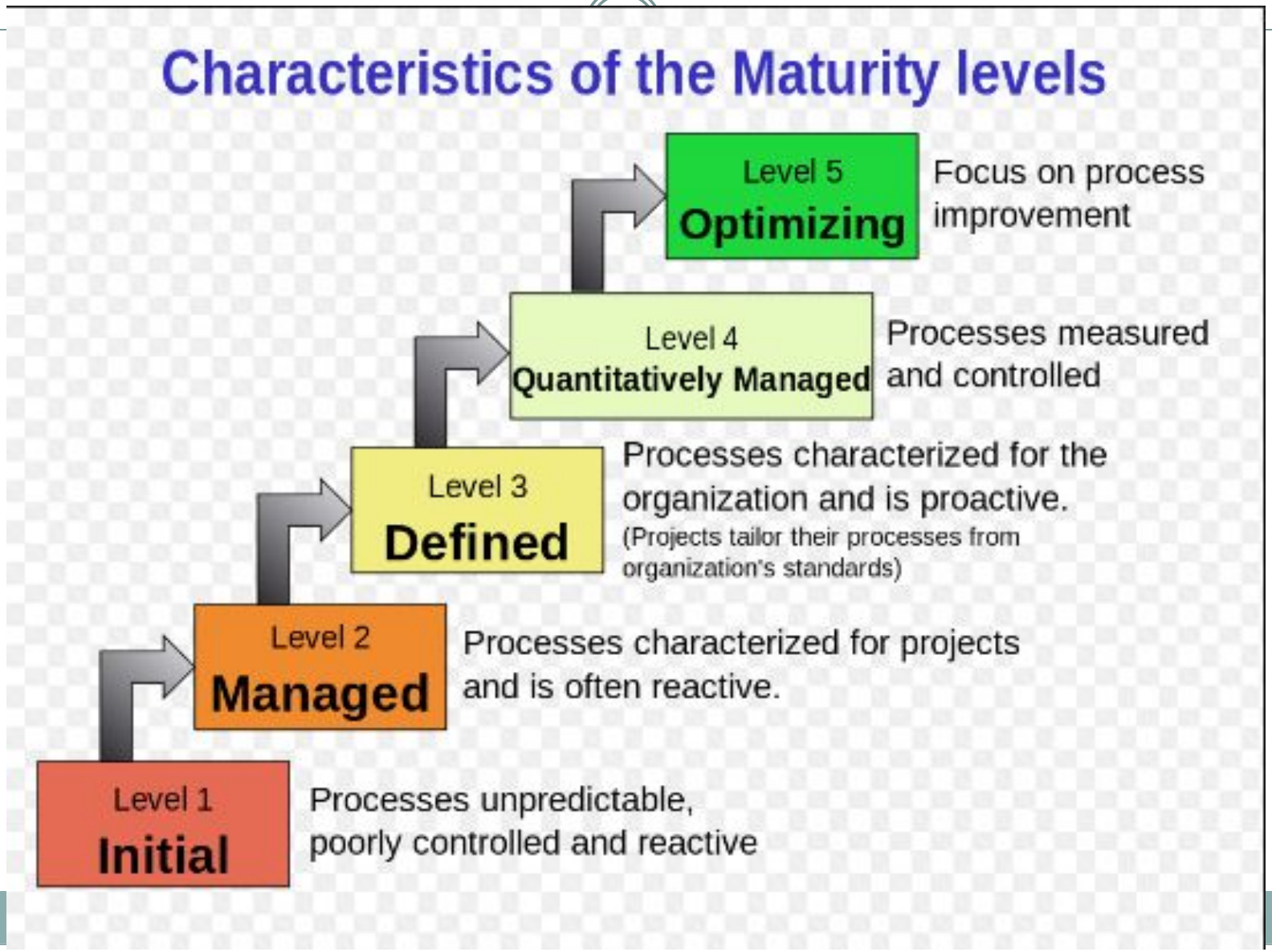
- Objective setting
  - Specific objectives for the phase are identified
- Risk assessment and reduction
  - Risks are assessed and activities put in place to reduce key risks
- Development and validation
  - A development model for the system is chosen which can be any of the generic models
- Planning
  - The project is reviewed and next phase of the spiral is planned

Spiral model usage

- Spiral model has been very influential in helping people think about iteration in software processes and introducing the risk-driven approach to development.

- In practice, however, the model is rarely used as published for practical software development.

CMMI: Capability Maturity Models Integrated



CMMI: Capability Maturity Models Integrated (cont.)

Level	Focus	Process Area	
5 Optimizing	Continuous Process Improvement	•Organizational Performance Management	•Causal Analysis & Resolution
4 Quantitatively Managed	Quantitative Management	•Organizational Process Performance	•Quantitative Project Management
3 Defined	Process Standardization	•Requirements Development •Technical Solutions •Product Integration •Verification •Validation •Organizational Process Focus	•Organizational Process Definition •Organizational Training •Integrated Project Management Risk Management •Decision Analysis & Resolution
2 Managed	Basic Project Management	•Requirements Management •Project Planning •Project Monitoring & Control •Supplier Agreement Management	•Measurement & Analysis •Process & Product Quality Assurance •Configuration Management
1 Initial			

Summary

- Iterative model - This model iterates requirements, design, build and test phases again and again for each requirement and builds up a system iteratively till the system is completely build.
- Incremental model - It is non integrated development model. This model divides work in chunks and one team can work on many chunks. It is more flexible.
- Spiral model - This model uses series of prototypes in stages, the development ends when the prototypes are developed into functional system. It is flexible model and used for large and complicated projects.
- Evolutionary model - It is more customer focused model. In this model the software is divided in small units which is delivered earlier to the customers.
- V-Model - It is more focused on testing. For every phase some testing activity are done.