M

Software Engineering (SDLC – Process Models)

Aug 2023

M

Types of Software Projects

Development	Maintenance	Conversion
• Full • Part	• Maintenance • Enhancements • Production Support	• Re-engineering • Migration • Porting

- Full Development - From requirement analysis to post project implementation and warranty
- Part Development – Partial Development
- Regular Maintenance - Revision or modification (Correct, adapt, or extend)
- Re-engineering – With intent of rebuilding existing system in whole or in part
- Migration - involves moving an existing applications from one platform to another
- Porting - Making existing source code available on more than one environments.

Systems Development Life Cycle (SDLC)

Life-Cycle Phases

System Concept Development

Defines the scope or boundary of the concepts. Includes Systems Boundary Document. Cost Benefit Analysis. Risk Management Plan and Feasibility Study.

Planning

Develops a Project Management Plan and other planning documents. Provides the basis for acquiring the resources needed to achieve a soulution.

Requirements Analysis

Analyses user needs and develops user requirements. Create a detailed Functional Requirements Document.

Design

Transforms detailed requirements into complete, detailed Systems Design Document Focuses on how to deliver the required functionality

Development

Converts a design into a complete information system Includes acquiring and installing systems environment; creating and testing databases preparing test case procedures; preparing test files, coding, compiling, refining programs; performing test readiness review and procurement activities.

Integration and Test

Demonstrates that developed system conforms to requirements as specified in the Functional Requirements Document. Conducted by Quality Assurance staff and users. Produces Test Analysis Reports.

Implementation

Includes implementation preparation, implementation of the system into a production environment, and resolution of problems identified in the Integration and Test Phases

Operations & Maintenance

Describes tasks to operate and maintain information systems in a production environment. includes Post-Implementation and In-Process Reviews.

Dispos

Describ end-of- activitie emphas given t proper prepara of data

M Process Models

A framework that describes the activities performed at each stage of a software development project

- Waterfall
- Incremental
- RAD
- V Shaped
- Prototyping
- Spiral
- Agile

Prescriptive

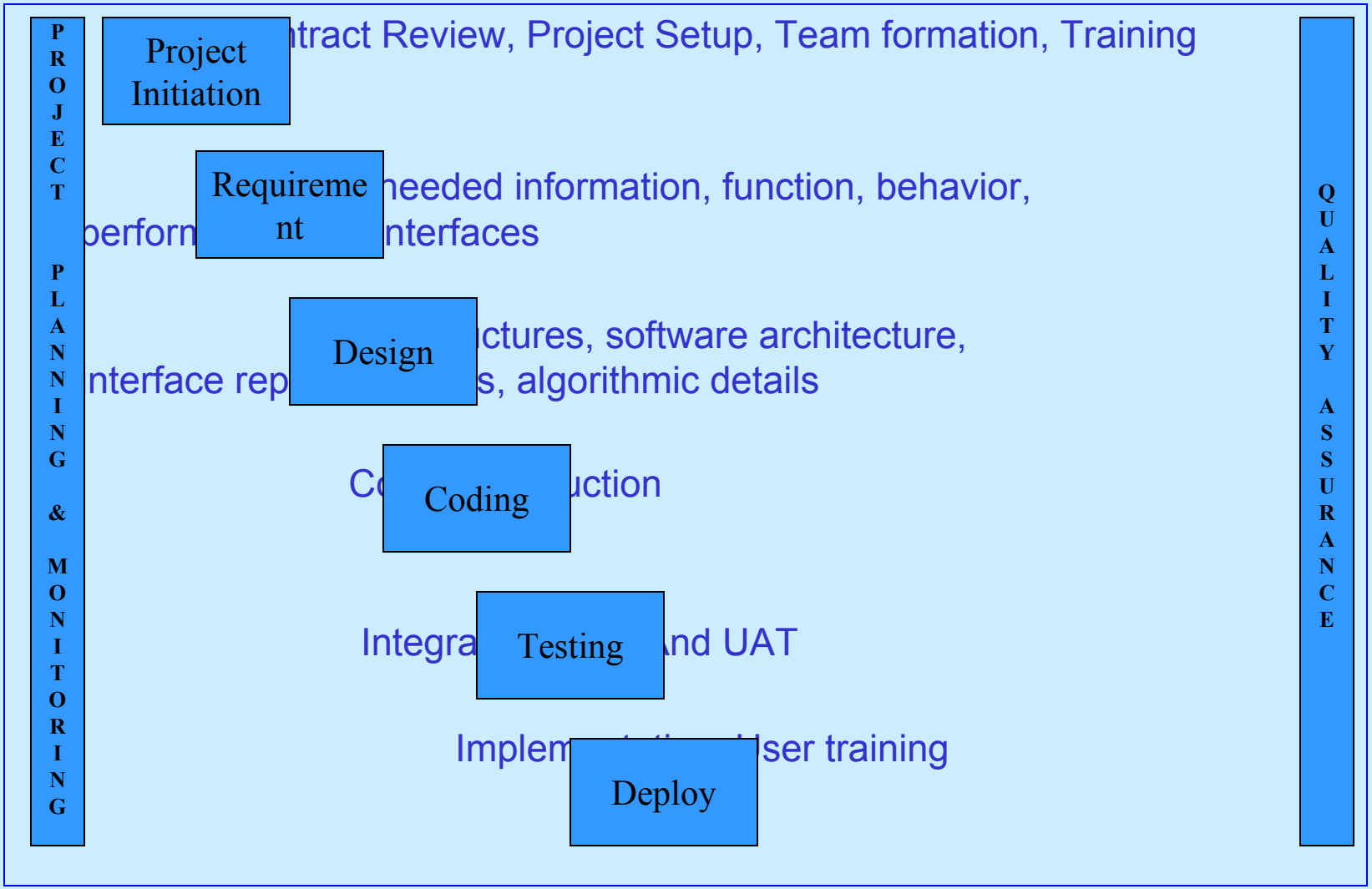
- ❖Define a distinct set of activities, actions, tasks, milestones, & work products
- ❖provides stability, control to an activity
- ❖populates a process frame work with explicit task set for s/w engineering actions

Evolutionary

- ❖combination of characteristics of conventional models, implemented in agile way

- Adaptive S/W Dev (ASD), Feature Driven (FDD), Extreme Prog., Rational Unify Process (RUP), Dynamic S/W Dev Method (DSDM)

Waterfall Model (Linear Sequential)



Waterfall Model…..

Strengths

- Easy to understand, easy to use
- Provides structure to inexperienced staff
- Milestones are well understood
- Sets requirements stability
- Good for management control (plan, staff, track)
- Works well when quality is more important than cost or schedule

Weaknesses

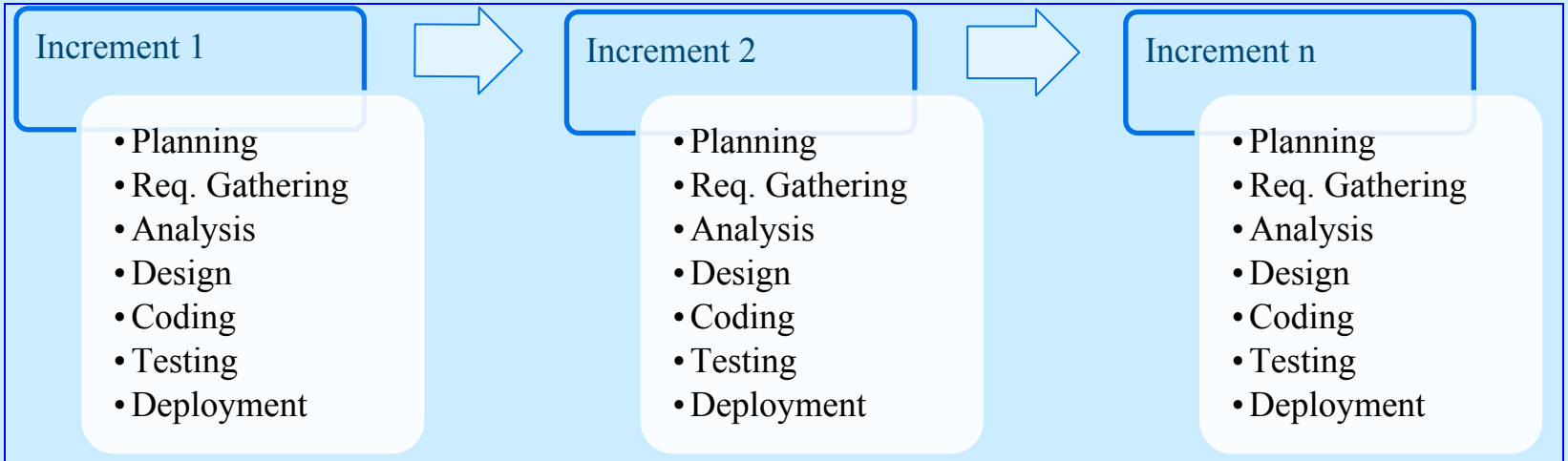
- All requirements must be known upfront
- Inhibits flexibility: Deliverables are considered frozen
- Can give a false impression of progress
- Does not reflect problem-solving nature of software development – iterations of phases
- Integration is one big bang at the end
- Little opportunity for customer to preview the system (until it may be too late)

Waterfall Model…..

When to Use?

- Requirements are very well known
- Product definition is stable
- Technology is understood
- New version of an existing product
- Porting an existing product to a new platform
- Quality is more important than cost & schedule

M Incremental Model



- Construct a partial implementation of a total system
- Then slowly add increased functionality
- The incremental model prioritizes requirements of the system and then implements them in groups.
- Each subsequent release of the system adds function to the previous release, until all designed functionality has been implemented.
- First increment is usually a core product which is built on top of, in the next increments - Each increment results in a finished product.

M Incremental Model…..

Strengths

- Develop high-risk or major functions first
- Each release delivers an operational product
- Customer can respond to each build
- Uses "divide and conquer" breakdown of tasks
- Lowers initial delivery cost
- Initial product delivery is faster
- Customers get important functionality early
- Risk of changing requirements is reduced

Weaknesses

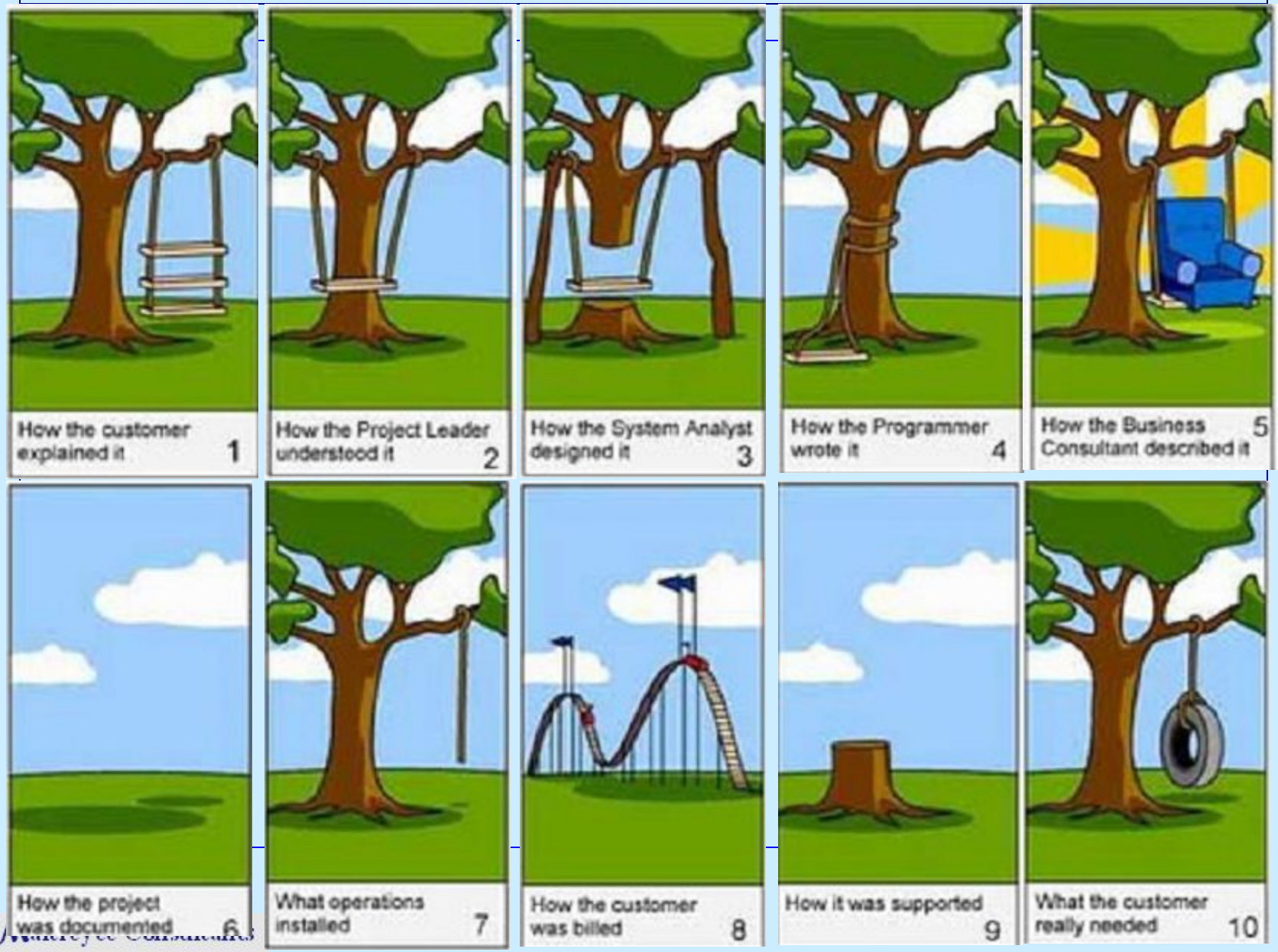
- Requires good planning and design
- Requires early definition of a complete and fully functional system to allow for the definition of increments
- Well-defined module interfaces are required (some will be developed long before others)
- Total cost of the complete system is not lower

Incremental Model…..

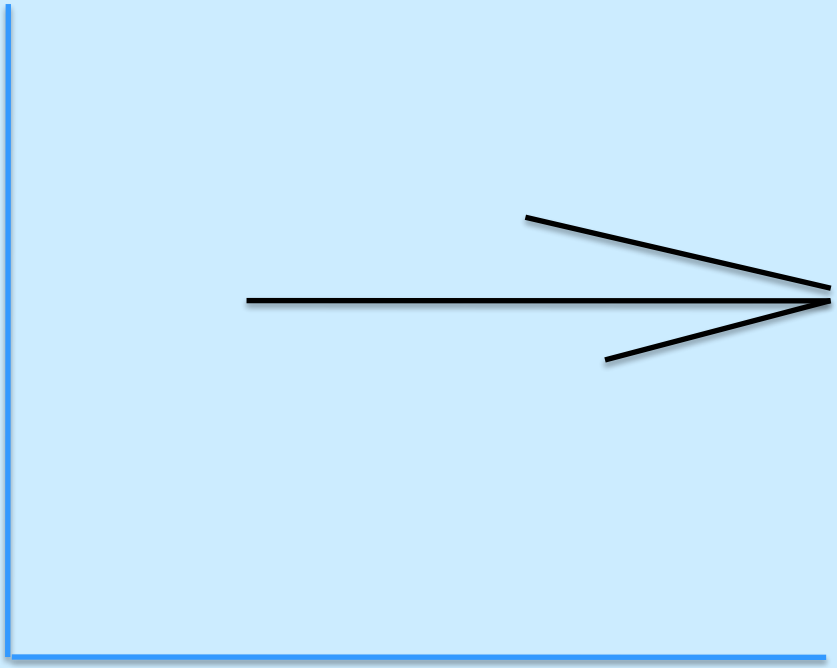
When to Use?

- Risk, funding, schedule, program complexity, or need for early realization of benefits.
- Most of the requirements are known up-front but are expected to evolve over time
- A need to get basic functionality to the market early
- On projects which have lengthy development schedules
- On a project with new technology

SDLC???


How the customer explained it 1
How the Project Leader understood it 2
How the System Analyst designed it 3
How the Programmer wrote it 4
How the Business Consultant described it 5
How the project was documented 6
What operations installed 7
How the customer was billed 8
How it was supported 9
What the customer really needed 10

Rapid Application Development



- Requirements planning phase (Discussion of business problems)
- User description phase – automated tools capture information from users
- Construction phase – productivity tools, such as code generators, screen generators, etc.)
- Cutover phase -- installation of the system, user acceptance testing and user training

Rapid Application Development…..

Strengths

- Reduced cycle time and improved productivity with fewer people means lower costs
- Time-box approach mitigates cost and schedule risk
- Customer involved throughout the complete cycle minimizes risk of not achieving customer satisfaction and business needs
- Focus moves from documentation to code (WYSIWYG).
- Uses modeling concepts to capture information about business, data, and processes

Weaknesses

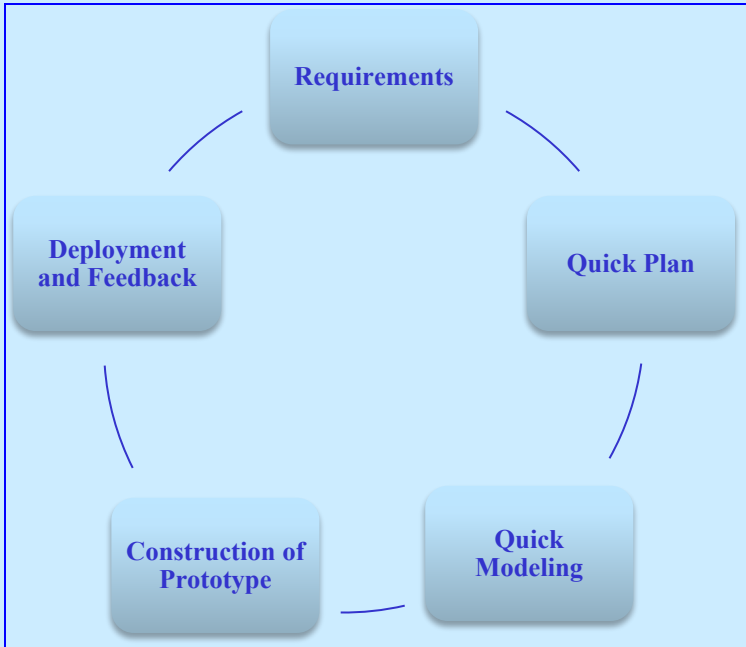
- Accelerated development process must give quick responses
- Risk of never achieving closure
- Hard to use with legacy systems
- Requires a system that can be modularized
- Developers and customers must be committed to rapid-fire activities in an abbreviated time frame.

Rapid Application Development…..

When to Use?

- Reasonably well-known requirements
- User involved throughout the life cycle
- Project can be time-boxed
- Functionality delivered in increments
- High performance not required
- Low technical risks
- System can be modularized

Prototyping Model



- Developers build a prototype during the requirements phase
- Prototype is evaluated by end users
- Users give corrective feedback
- Developers further refine the prototype
- When the user is satisfied, the prototype code is brought up to the standards needed for a final product

Prototyping Model…..

Strengths

- Customers can “see” the system requirements as they are being gathered
- Developers learn from customers
- A more accurate end product
- Unexpected requirements accommodated
- Allows for flexible design and development
- Steady, visible signs of progress produced
- Interaction with the prototype stimulates awareness of additional needed functionality

Weaknesses

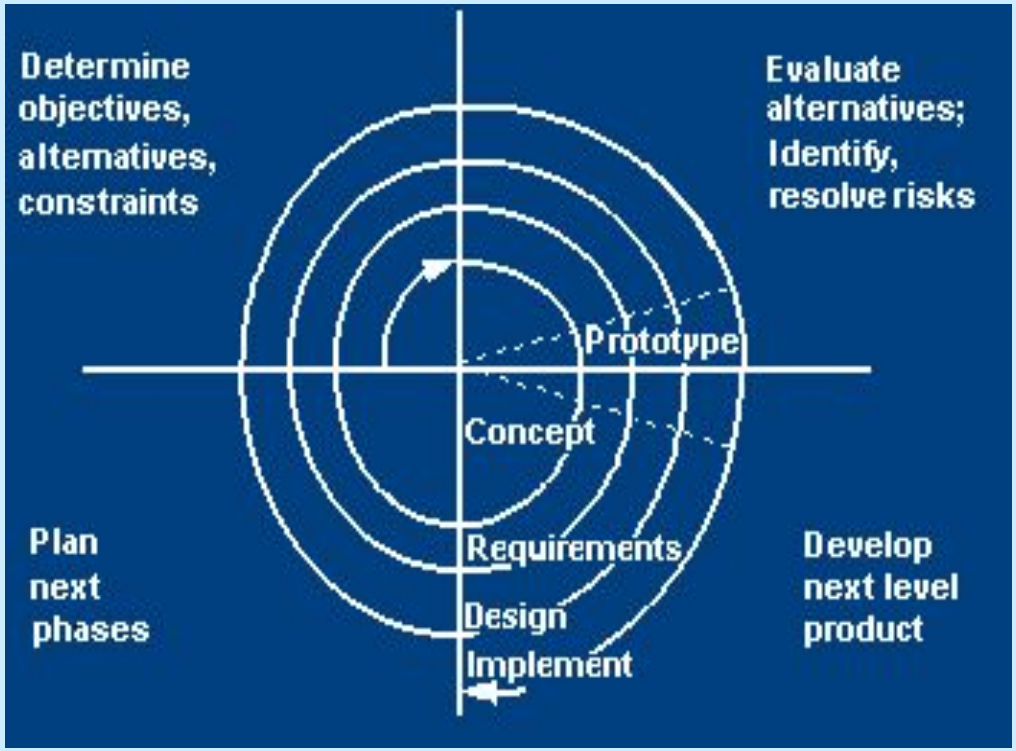
- Tendency to abandon structured program development for “code-and-fix” development
- Bad reputation for “quick-and-dirty” methods
- Overall maintainability may be overlooked
- The customer may want the prototype delivered
- Process may continue forever (scope creep)

Prototyping Model…..

When to Use?

- Requirements are unstable or have to be clarified
- As the requirements clarification stage of a waterfall model
- Develop user interfaces
- Short-lived demonstrations
- New, original development
- With the analysis and design portions of object-oriented development.
- When customer has legitimate need but clueless about its details

Spiral Model



- Adds risk analysis, and 4gl RAD prototyping to the waterfall model
- Each cycle involves the same sequence of steps as the waterfall process model

- Introduced by B Boehm.
- The model is spiral starts in middle & revisits the basic tasks

M

Spiral Model…..

Quadrants of Spiral Model

Determine Objectives, Alternatives & Constraints	Evaluate Alternative Identify & Resolve Risk	Develop Next Level Product	Plan Next Phase
• Objectives: functionality, performance, hardware/software interface, critical success factors, etc. • Alternatives: build, reuse, buy, sub-contract, etc. • Constraints: cost, schedule, interface, etc.	• Study alternatives relative to objectives and constraints • Identify risks (lack of experience, new technology, tight schedules, poor process, etc. • Resolve risks (evaluate if money could be lost by continuing system development	• Create a design • Review design • Develop code • Inspect code • Test product	• Develop project plan • Develop configuration management plan • Develop a test plan • Develop an installation plan

Spiral Model…..

Strengths

- Provides early indication of insurmountable risks, without much cost
- Users see the system early because of rapid prototyping tools
- Critical high-risk functions are developed first
- The design does not have to be perfect
- Users can be closely tied to all lifecycle steps
- Early and frequent feedback from users
- Cumulative costs assessed frequently

Weaknesses

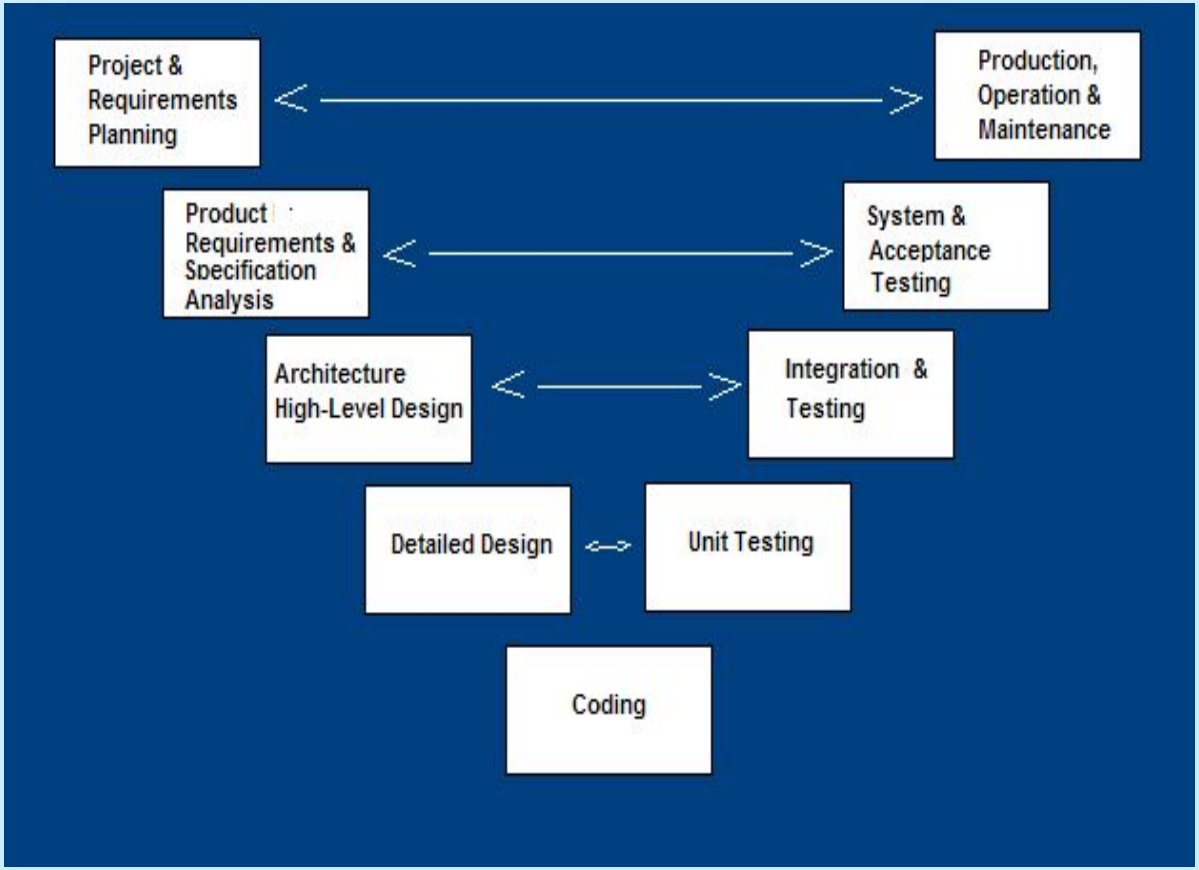
- Time spent for evaluating risks too large for small or low-risk projects
- Time spent planning, resetting objectives, doing risk analysis and prototyping may be excessive
- The model is complex
- Risk assessment expertise is required
- Spiral may continue indefinitely
- Developers must be reassigned during non-development phase activities
- May be hard to define objective, verifiable milestones that indicate readiness to proceed through the next iteration

Spiral Model…..

When to Use?

- When creation of a prototype is appropriate
- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

M V Shaped Model



- A variant of the Waterfall that emphasizes the verification and validation
- Testing of the product is planned in parallel with a corresponding phase of development

M V Shaped Model…..

Activities

- Project and Requirements Planning – allocate resources
- Product Requirements and Specification Analysis – complete specification of the software system
- Architecture or High-Level Design – defines how software functions fulfill the design
- Detailed Design – develop algorithms for each architectural component
- Production, operation and maintenance – provide for enhancement and corrections
- System and acceptance testing – check the entire software system in its environment
- Integration and Testing – check that modules interconnect correctly
- Unit testing – check that each module acts as expected
- Coding – transform algorithms into software

V Shaped Model…..

Strengths

- Emphasize planning for verification and validation of the product in early stages of product development
- Each deliverable must be testable
- Project management can track progress by milestones
- Easy to use

Weaknesses

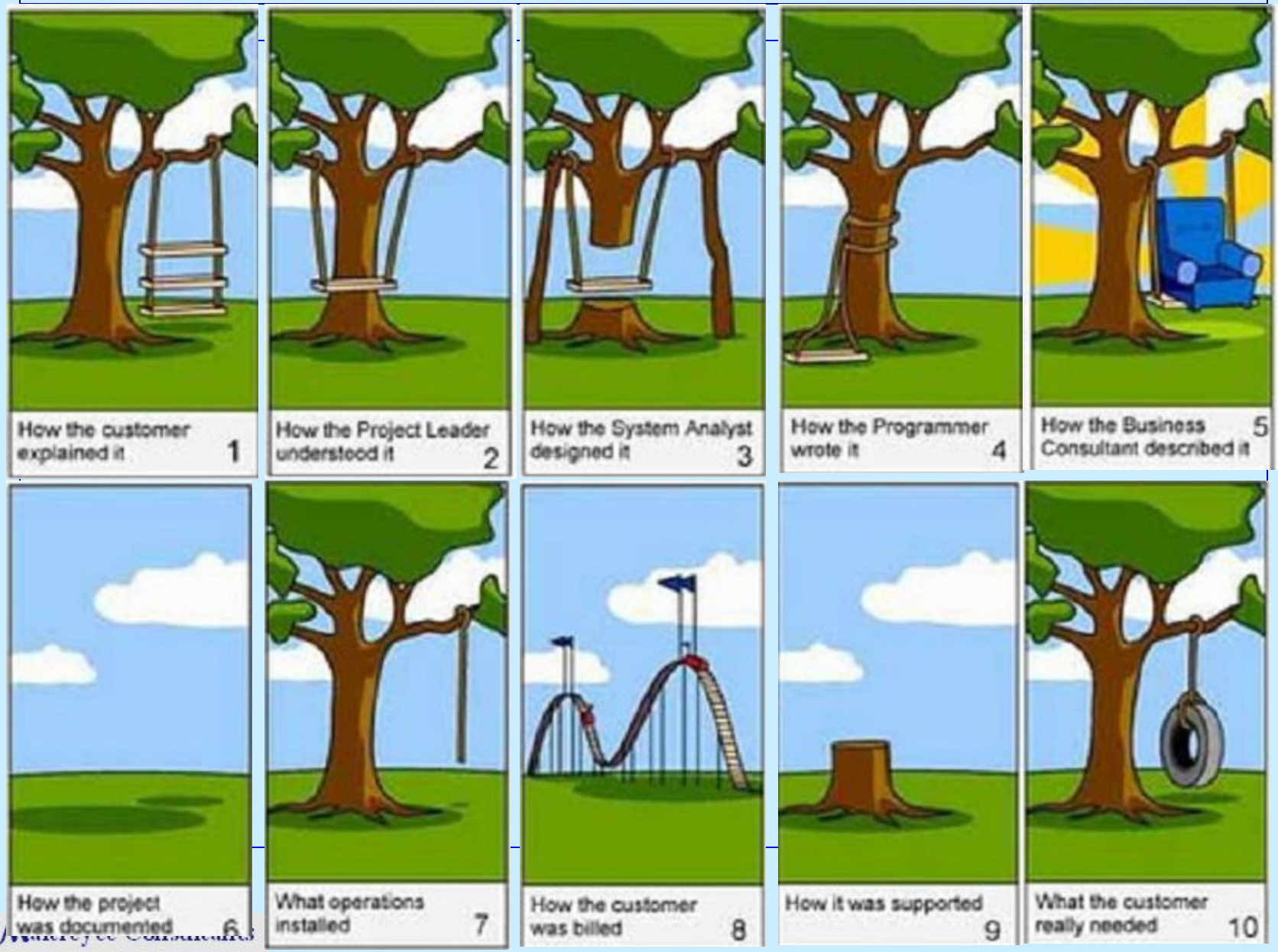
- Does not easily handle concurrent events
- Does not handle iterations or phases
- Does not easily handle dynamic changes in requirements
- Does not contain risk analysis activities

V Shaped Model…..

When to Use?

- Excellent choice for systems requiring high reliability – hospital patient control applications
- All requirements are known up-front
- When it can be modified to handle changing requirements beyond analysis phase
- Solution and technology are known

SDLC???


How the customer explained it 1
How the Project Leader understood it 2
How the System Analyst designed it 3
How the Programmer wrote it 4
How the Business Consultant described it 5
How the project was documented 6
What operations installed 7
How the customer was billed 8
How it was supported 9
What the customer really needed 10

Q & A

Any Question?