Introduction to software engineering

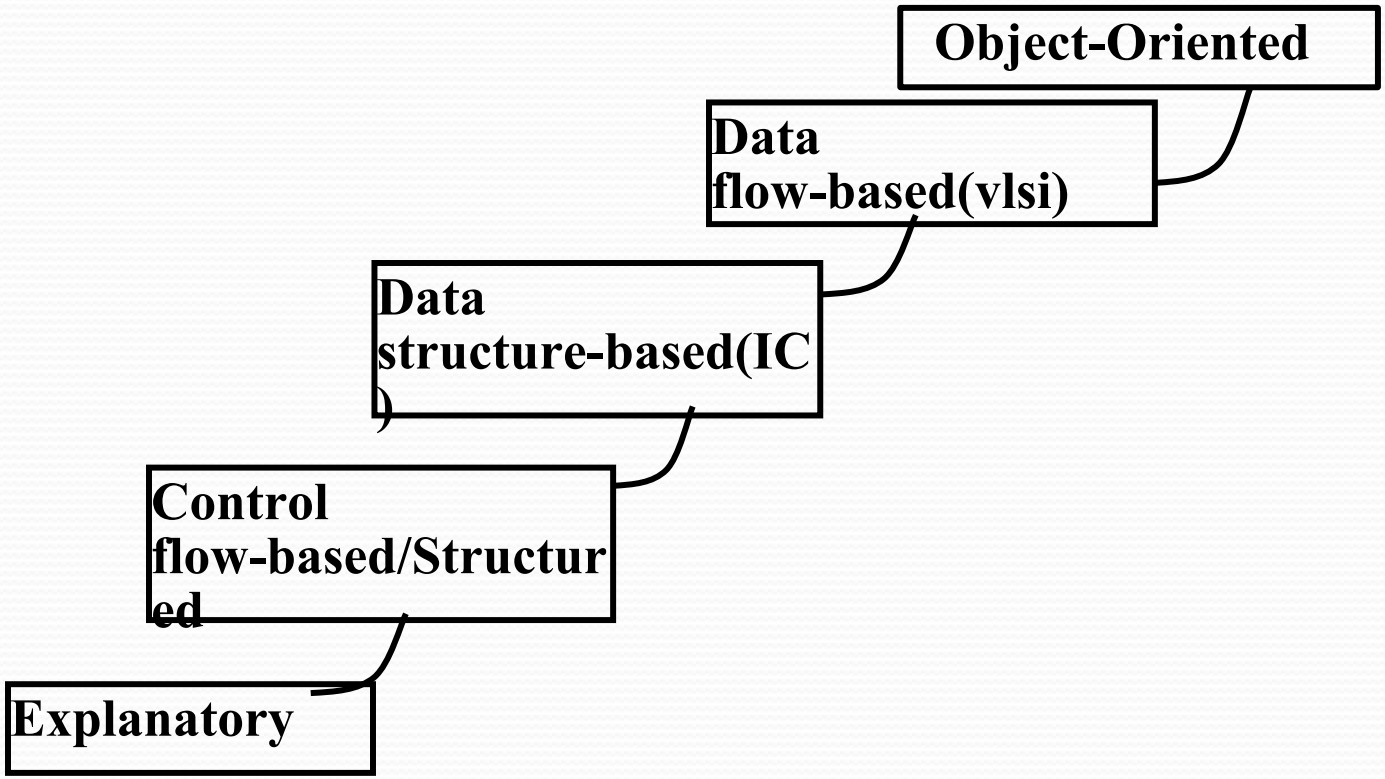
Objectives

- Software Engineering
- To introduce software process
- Software Engineering
- SDLC Cycle
- SDLC models

Software Engineering

- Ability to solve complex programming problems:
  - How to break large projects into smaller and manageable parts?
  - Handling complexity in a software development problem is the main theme of software engineering discipline.
  - 2 techniques: abstraction and decomposition
  - Learn techniques of: specification, design, interface development, testing, project management, etc.

Evolution of Design Techniques



Evolution of Other Software Engineering Techniques

- life cycle models,
- specification techniques,
- project management techniques,
- testing techniques,
- debugging techniques,
- quality assurance techniques,
- software measurement techniques,
- CASE tools, etc.

Differences Between the Exploratory Style and Modern Software Development Practices

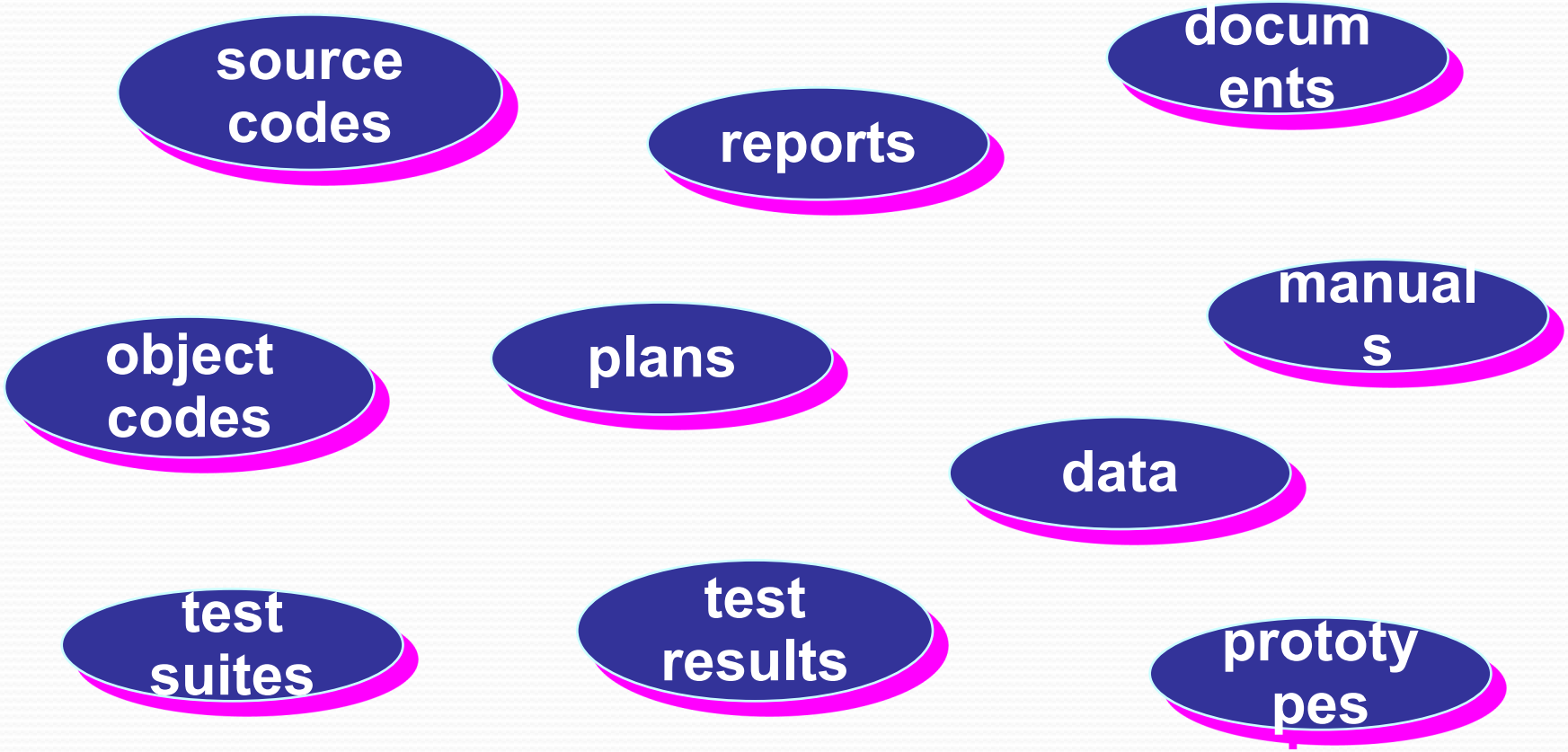
- Emphasis has shifted
- from error correction to error prevention.
- Modern practices emphasize
- detection of errors as close to their point of introduction as possible.

Differences Between the Exploratory Style and Modern Software Development Practices (CONT.)

- In exploratory style,
  - errors are detected only during testing,
- Now,
  - focus is on detecting as many errors as possible in each phase of development.
- A lot of effort and attention is now being paid to:
  - requirements specification.
- Also, now there is a distinct design phase:
  - standard design techniques are being used.

Software Product

is a product designated for delivery to the user



Programs versus Software Products

Usually small in size	Large
Author himself is sole user	Large number of users
Single developer	Team of developers
Lacks proper user interface	Well-designed interface
Lacks proper documentation	Well documented & user-manual prepared
Ad hoc development.	Systematic development

The software process

- A structured set of activities required to develop a software system
  - Specification;
- Design;
  - 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 Life Cycle

- Software life cycle (or software process):
  - series of identifiable stages that a software product undergoes during its life time:
    - Feasibility study
    - requirements analysis and specification,
    - design,
    - coding,
    - testing
    - maintenance.
    - Each of these stages is called ‘life cycle phase’.

Life Cycle Model

- A software life cycle model (or process model):
  - a descriptive and diagrammatic model of software life cycle:
  - identifies all the activities required for product development,
  - establishes a precedence ordering among the different activities,
  - Divides life cycle into phases.

Life Cycle Model (CONT.)

- Several different activities may be carried out in each life cycle phase.
  - For example, the design stage might consist of:
    - structured analysis activity followed by
    - structured design activity.

Why Model Life Cycle ?

- A written description:
  - forms a common understanding of activities among the software developers.
  - helps in identifying inconsistencies, redundancies, and omissions in the development process.
  - Helps in tailoring a process model for specific projects.

Why Model Life Cycle ?

- Processes are tailored for special projects.
  - A documented process model
    - helps to identify where the tailoring is to occur.
- life cycle model:
  - defines entry and exit criteria for every phase.
  - A phase is considered to be complete:
    - only when all its exit criteria are satisfied.

Generic software process models

- The waterfall model
  - Separate and distinct phases of specification and development.
- Evolutionary development
  - Specification, development and validation are interleaved.
- Component-based software engineering
  - The system is assembled from existing components.
- RAD
- Agile developments.

Various stages in software process

Feasibility Study

- Main aim of feasibility study: determine whether developing the product
  - financially worthwhile
  - technically feasible.
- First roughly understand what the customer wants:
  - different data which would be input to the system,
  - processing needed on these data,
  - output data to be produced by the system,
  - various constraints on the behavior of the system.

Activities during Feasibility Study

- Work out an overall understanding of the problem.
- Formulate different solution strategies.
- Examine alternate solution strategies in terms of:
  - resources required,
  - cost of development, and
  - development time.

Activities during Feasibility Study

- Perform a cost/benefit analysis:
  - to determine which solution is the best.
  - you may determine that none of the solutions is feasible due to:
    - high cost,
    - resource constraints,
    - technical reasons.

Requirements Analysis and Specification

- Aim of this phase:
  - understand the exact requirements of the customer,
  - document them properly.
  - Consists of two distinct activities:
  - requirements gathering and analysis
  - requirements specification.

Goals of Requirements Analysis

- Collect all related data from the customer:
  - analyze the collected data to clearly understand what the customer wants,
  - find out any inconsistencies and incompleteness in the requirements,
  - resolve all inconsistencies and incompleteness.

Requirements Gathering

- Gathering relevant data:
  - usually collected from the end-users through interviews and discussions.
  - For example, for a business accounting software:
    - interview all the accountants of the organization to find out their requirements.

Requirements Analysis (CONT.)

- The data you initially collect from the users:
  - would usually contain several contradictions and ambiguities:
  - each user typically has only a partial and incomplete view of the system.

Requirements Analysis (CONT.)

- Ambiguities and contradictions:
  - must be identified
  - resolved by discussions with the customers.
- Next, requirements are organized:
  - into a Software Requirements Specification (SRS) document.
- Engineers doing requirements analysis and specification:
  - are designated as analysts.

Design

- Design phase transforms requirements specification:
  - into a form suitable for implementation in some programming language.
- In technical terms:
  - during design phase, software architecture is derived from the SRS document.
- Two design approaches:
  - traditional approach,
  - object oriented approach.

Traditional Design Approach

- Consists of two activities:
  - Structured analysis
  - Structured design

Structured Analysis Activity

- Identify all the functions to be performed.
- Identify data flow among the functions.
- Decompose each function recursively into sub-functions.
  - Identify data flow among the subfunctions as well.

Structured Analysis (CONT.)

- Carried out using Data flow diagrams (DFDs).
- After structured analysis, carry out structured design:
  - architectural design (or high-level design)
  - detailed design (or low-level design).

Structured Design

- High-level design:
  - decompose the system into modules,
  - represent invocation relationships among the modules.
- Detailed design:
  - different modules designed in greater detail:
    - data structures and algorithms for each module are designed.

Object Oriented Design

- First identify various objects (real world entities) occurring in the problem:
  - identify the relationships among the objects.
  - For example, the objects in a pay-roll software may be:
    - employees,
    - managers,
    - pay-roll register,
    - Departments, etc.

Object Oriented Design (CONT.)

- Object structure
  - further refined to obtain the detailed design.
- OOD has several advantages:
  - lower development effort,
  - lower development time,
  - better maintainability.

Implementation

- Purpose of implementation phase (aka coding and unit testing phase):
  - translate software design into source code.
- During the implementation phase:
  - each module of the design is coded,
  - each module is unit tested
    - tested independently as a stand alone unit, and debugged,
  - each module is documented.

Implementation (CONT.)

- The purpose of unit testing:
  - test if individual modules work correctly.
- The end product of implementation phase:
  - a set of program modules that have been tested individually.

Integration and System Testing

- Different modules are integrated in a planned manner:
  - modules are almost never integrated in one shot.
  - Normally integration is carried out through a number of steps.
- During each integration step,
  - the partially integrated system is tested.

Integration and System Testing

System Testing

- After all the modules have been successfully integrated and tested:
  - system testing is carried out.
- Goal of system testing:
  - ensure that the developed system functions according to its requirements as specified in the SRS document.

Maintenance

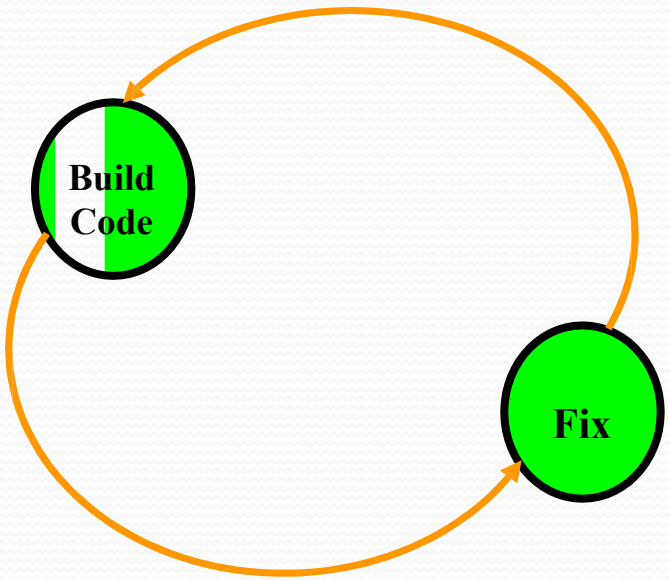
- Maintenance of any software product:
  - requires much more effort than the effort to develop the product itself.
  - development effort to maintenance effort is typically 40:60.

Maintenance (CONT.)

- Corrective maintenance:
  - Correct errors which were not discovered during the product development phases.
- Perfective maintenance:
  - Improve implementation of the system
  - enhance functionalities of the system.
- Adaptive maintenance:
  - Port software to a new environment,
    - e.g. to a new computer or to a new operating system.

Build & Fix Model

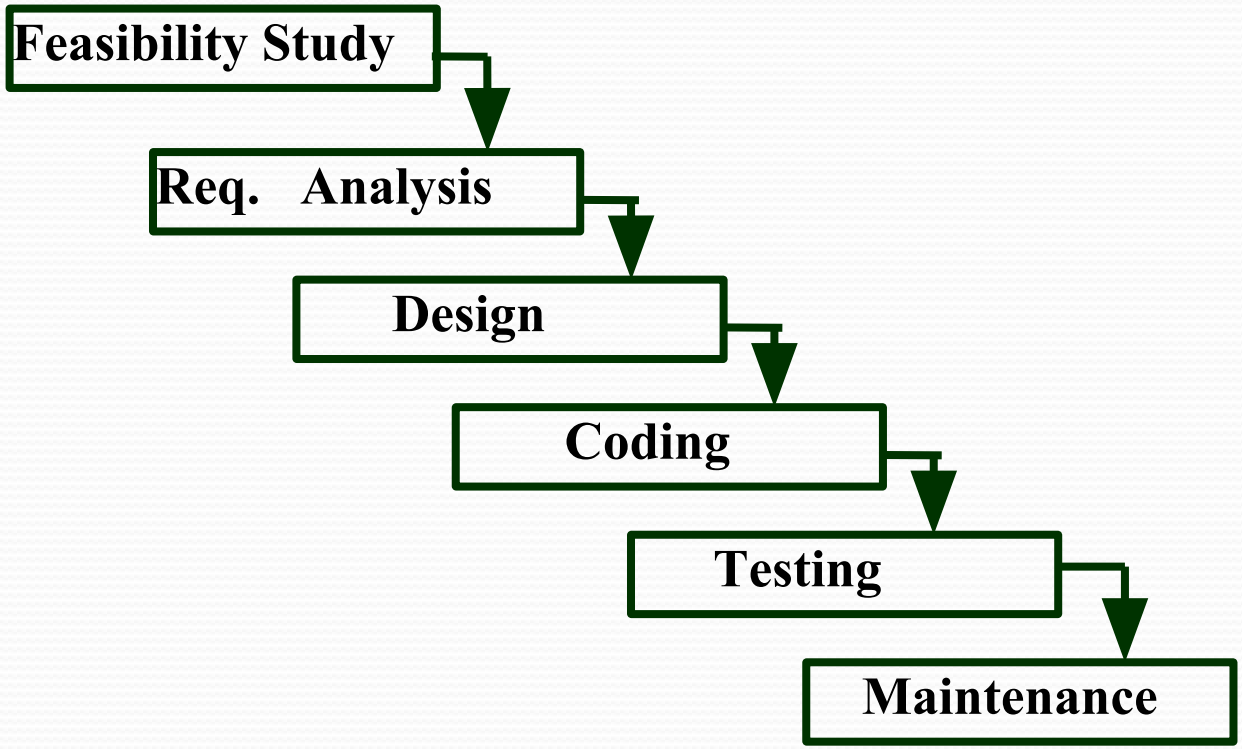
- Product is constructed without specifications or any attempt at design
- Adhoc approach and not well defined
- Simple two phase model



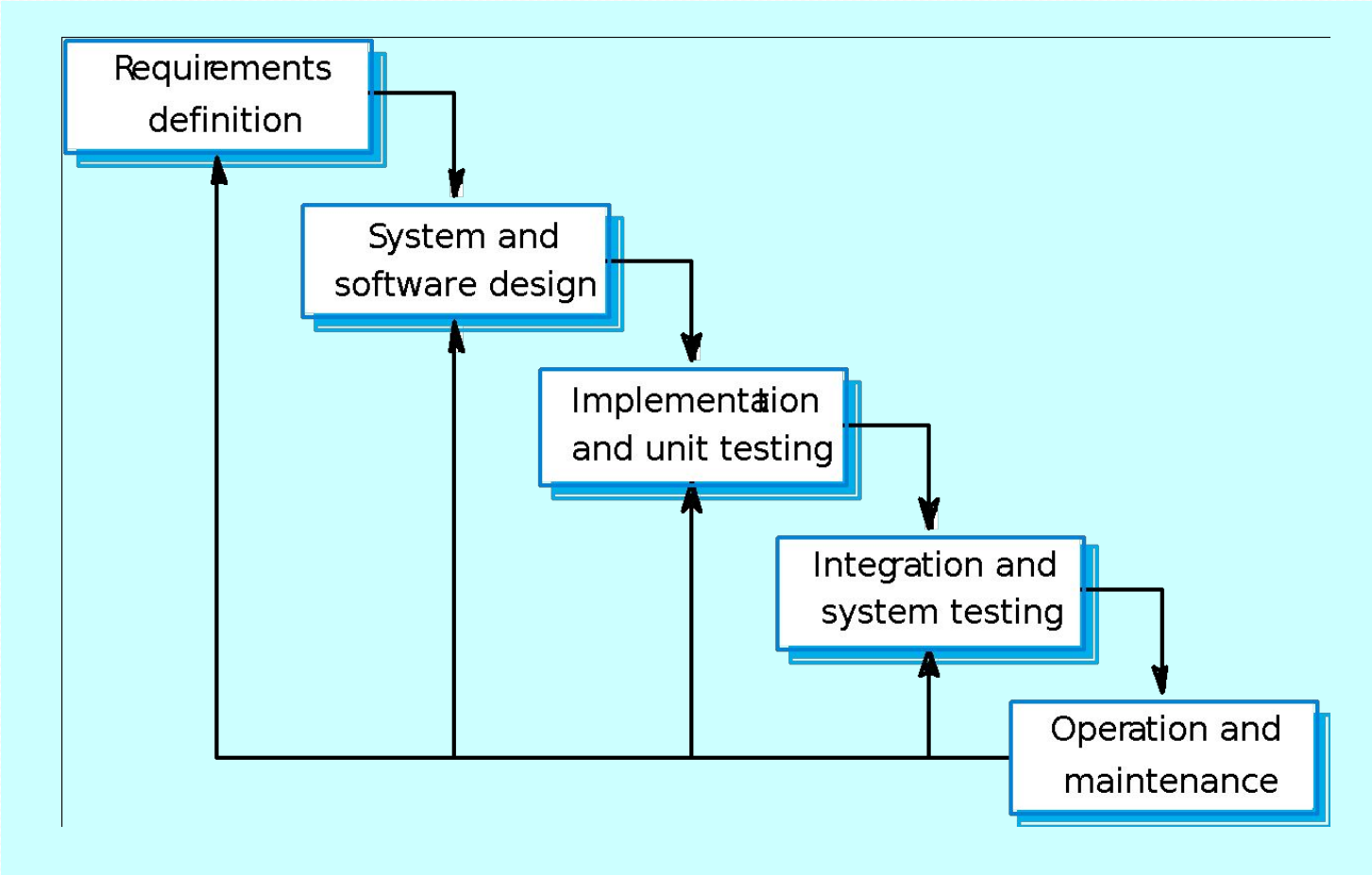
Build & Fix Model

- Suitable for small programming exercises of 100 or 200 lines
- Unsatisfactory for software for any reasonable size
- Code soon becomes unfixable & unenhanceable
- No room for structured design
- Maintenance is practically not possible

Classical Waterfall Model



Waterfall model



Waterfall 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

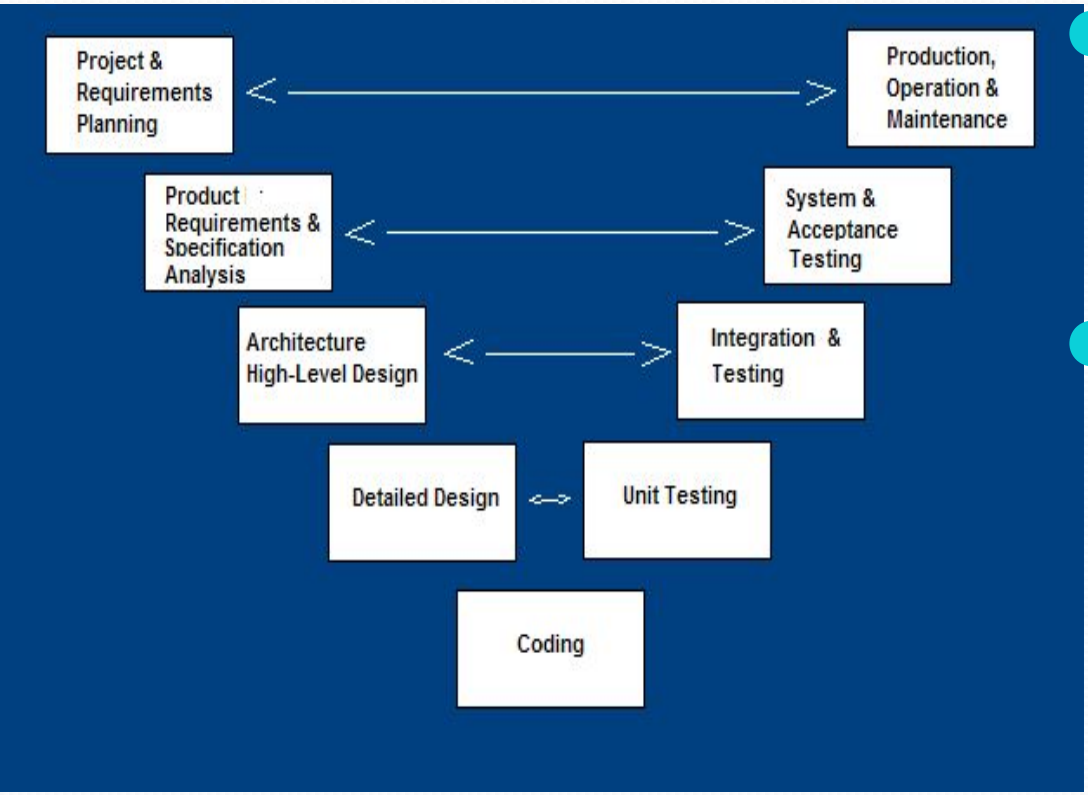
Waterfall Deficiencies

- All requirements must be known upfront
- Deliverables created for each phase are considered frozen – inhibits flexibility
- 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)

When to use the Waterfall Model

- 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.
- High risk for new systems because of specification and design problems.
- Low risk for well-understood developments using familiar technology.

V-Shaped SDLC Model



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

V-Shaped 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

V-Shaped 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

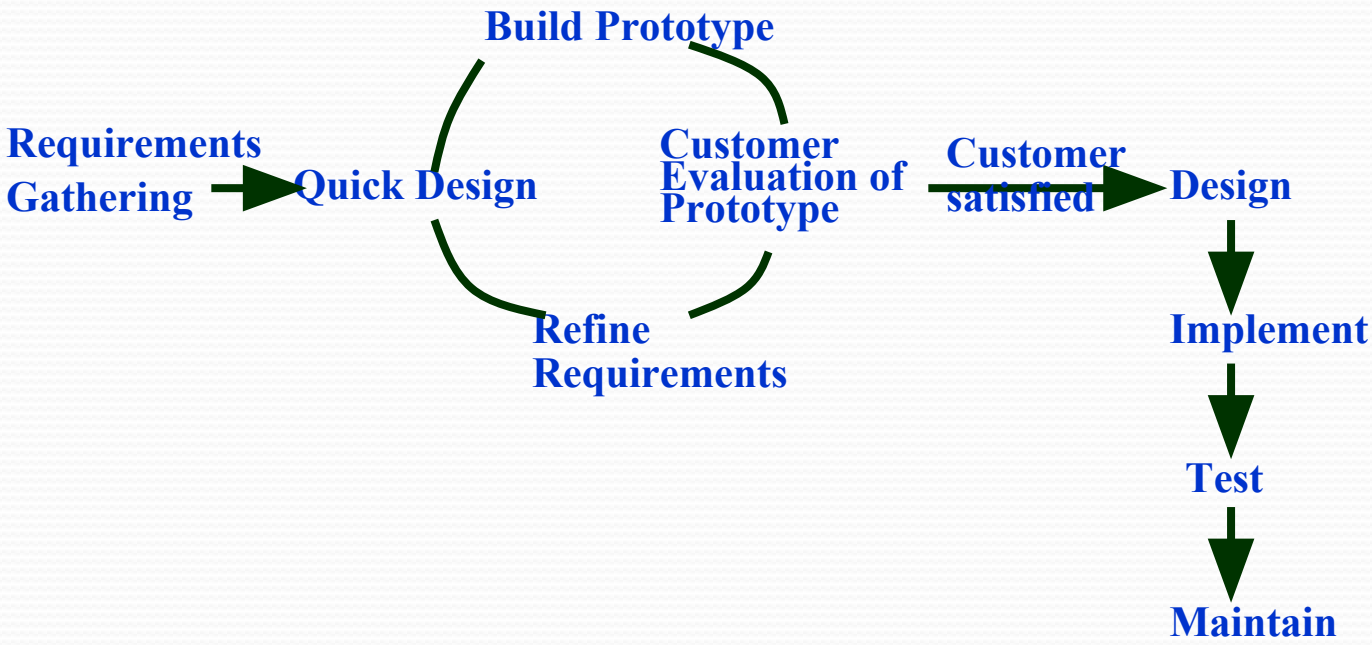
Prototyping Model

- Start with approximate requirements.
- Carry out a quick design.
- Prototype model is built using several short-cuts:
- Short-cuts might involve using inefficient, inaccurate, or dummy functions.
- A function may use a table look-up rather than performing the actual computations.

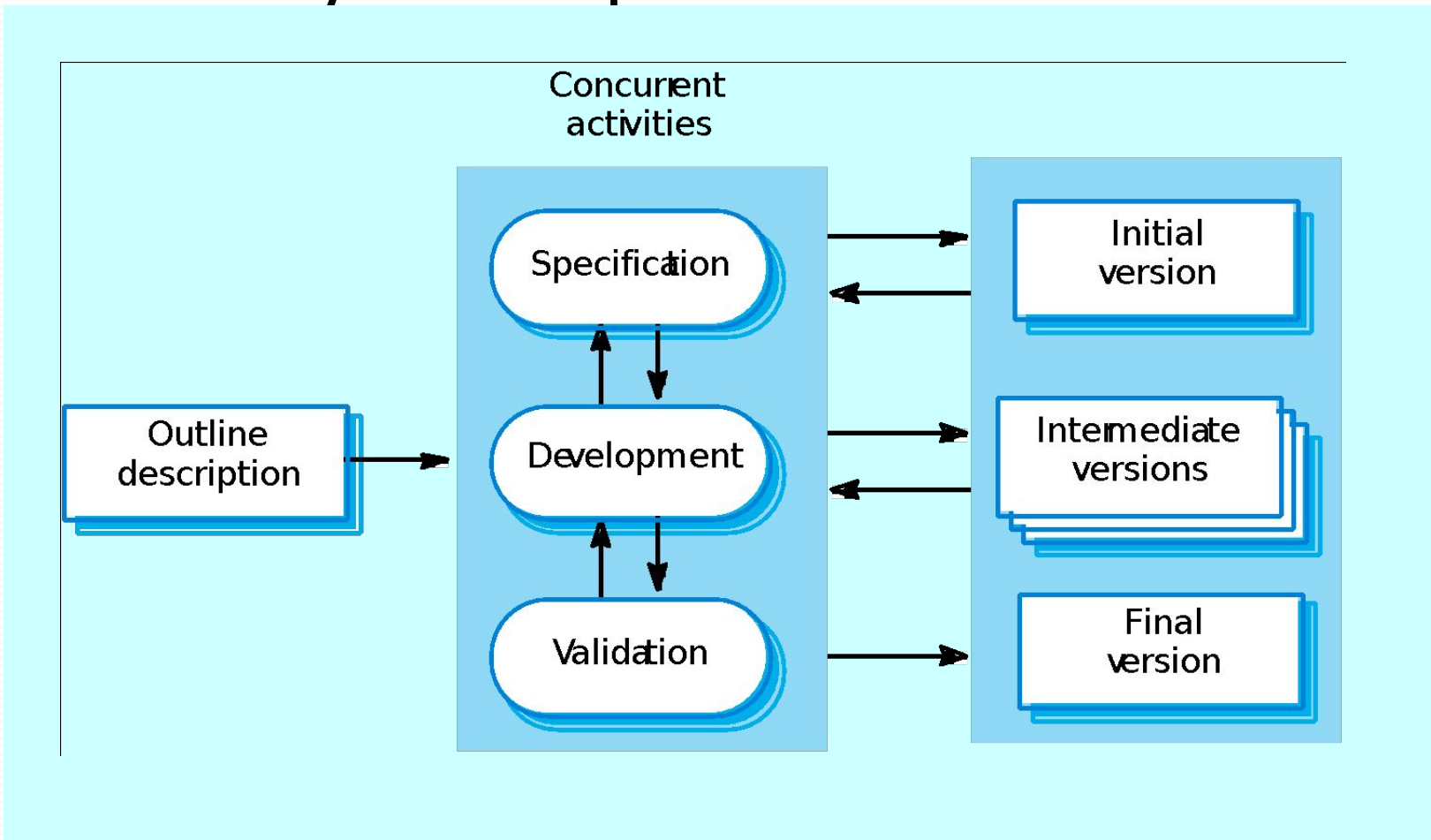
Prototyping Model (CONT.)

- The developed prototype is submitted to the customer for his evaluation:
- Based on the user feedback, requirements are refined.
- This cycle continues until the user approves the prototype.
- The actual system is developed using the classical waterfall approach.

Prototyping Model (CONT.)



Evolutionary development



Evolutionary development

- Exploratory development
- Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements and add new features as proposed by the customer.
- Throw-away prototyping
- Objective is to understand the system requirements. Should start with poorly understood requirements to clarify what is really needed.

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

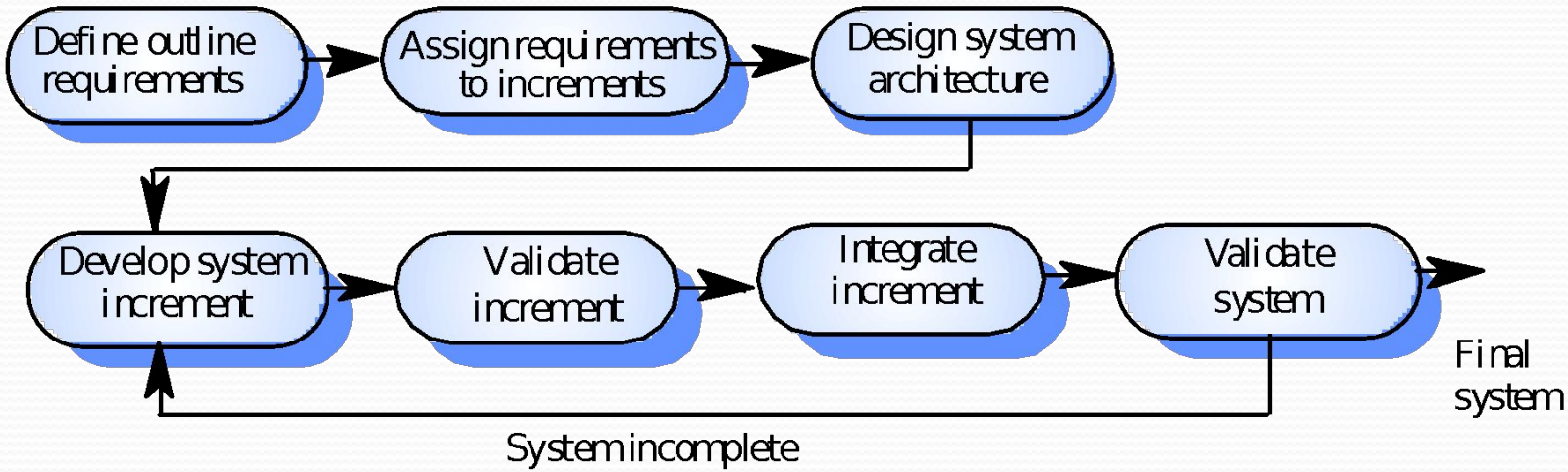
Evolutionary development

- Problems
- Lack of process visibility;
- Systems are often poorly structured;
- Special skills (e.g. in languages for rapid prototyping) may be required.
- Applicability
- For small or medium-size interactive systems;
- For parts of large systems (e.g. the user interface);
- For short-lifetime systems.

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).
- For short-lifetime systems.

Incremental means adding, iterative means reworking

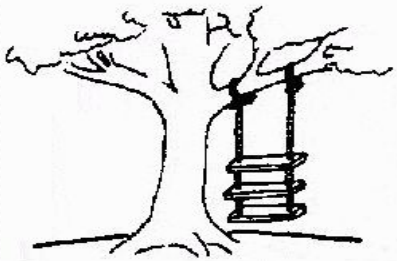
Increment	Iterate
fundamentally means *“add .”onto*	fundamentally means *.””“change*
repeating the process on a *new* .section of work	repeating the process on the *same* section of work
repeat the process (, design, ,implement, evaluate)	repeat the process (, design, ,implement, evaluate)

Incremental & iterative - summary

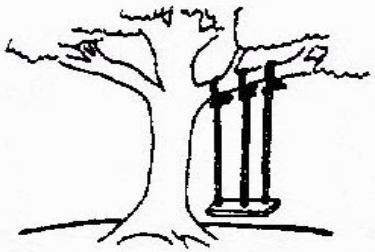
- Iterative model - This model iterates requirements, design, build and test phases again and again for each requirement and builds up a system iteratively till the system is completely build.
- Incremental model - It is non integrated development model. This model divides work in chunks and one team 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.

The Rapid Application Development (RAD) Model

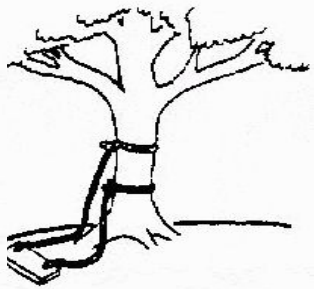
- Developed by IBM in 1980
- User participation is essential



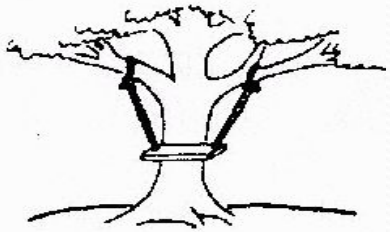
The requirements specification was defined like this



The developers understood it in that way



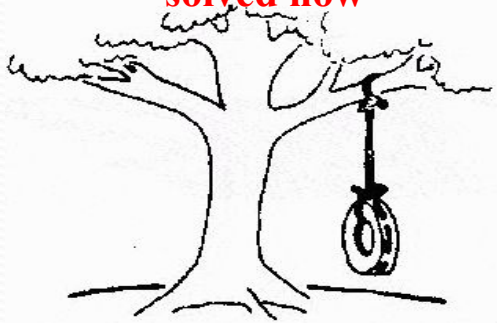
This is how the problem was solved before.



This is how the problem is solved now

That is the program after

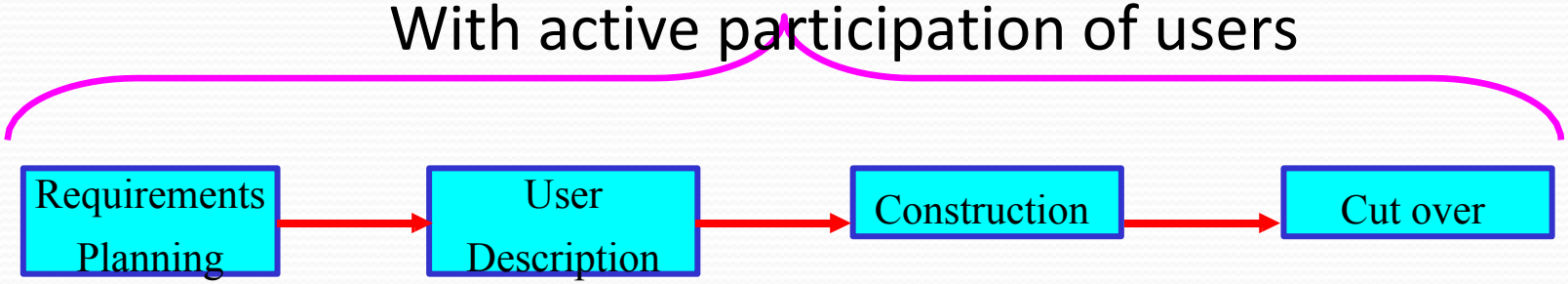
This is how the program is described by marketing department



This, in fact, is what the customer wanted …

The Rapid Application Development (RAD) Model

- Build a rapid prototype
- Give it to user for evaluation & obtain feedback
- Prototype is refined



The Rapid Application Development (RAD) Model

Not an appropriate model in the absence of user participation.

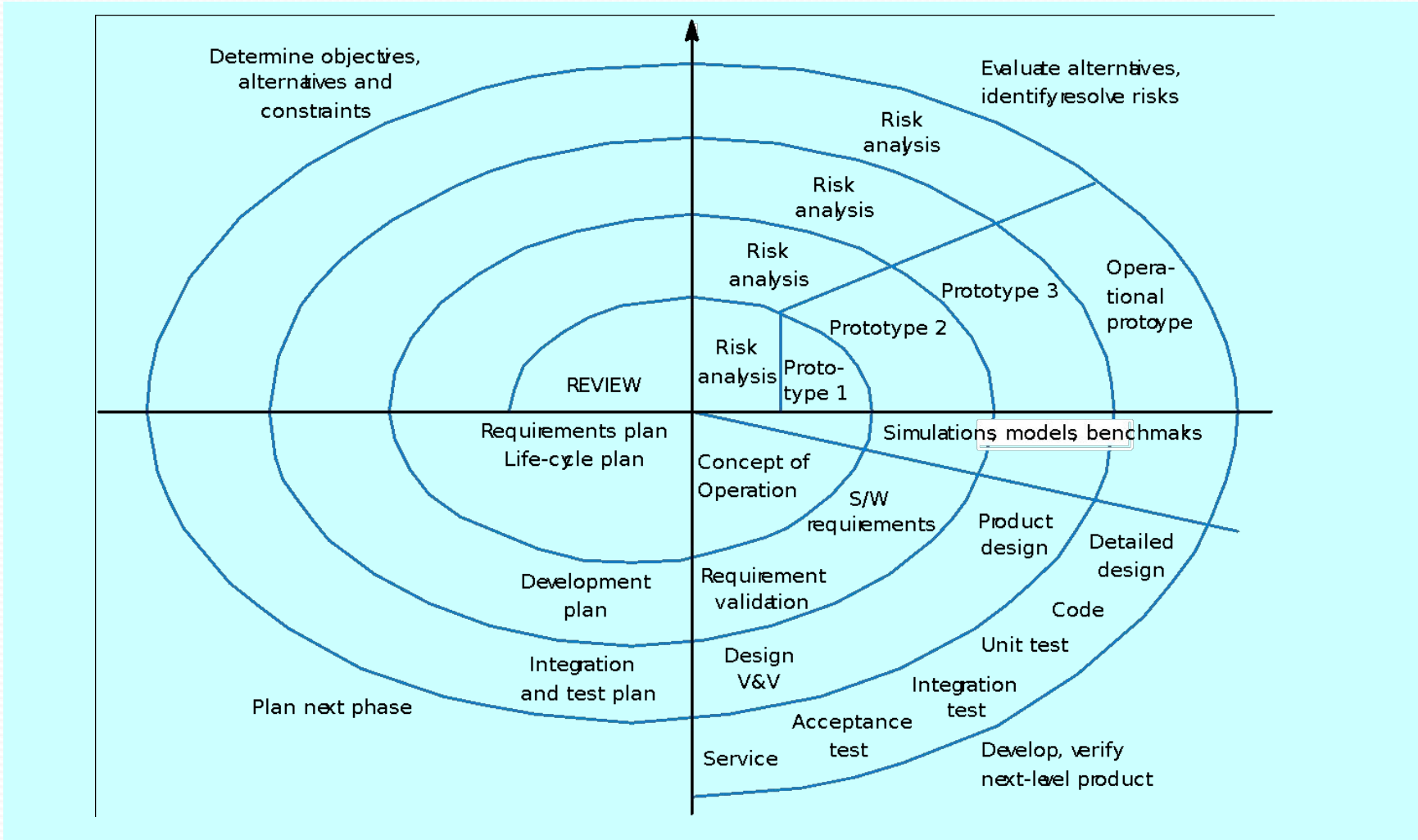
Reusable components are required to reduce development time.

Highly specialized & skilled developers are required and such developers are not easily available.

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 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

Spiral Model 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

When to use Spiral Model

- 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)

Agile Methods

- Result from dissatisfaction with the overheads involved in design methods.

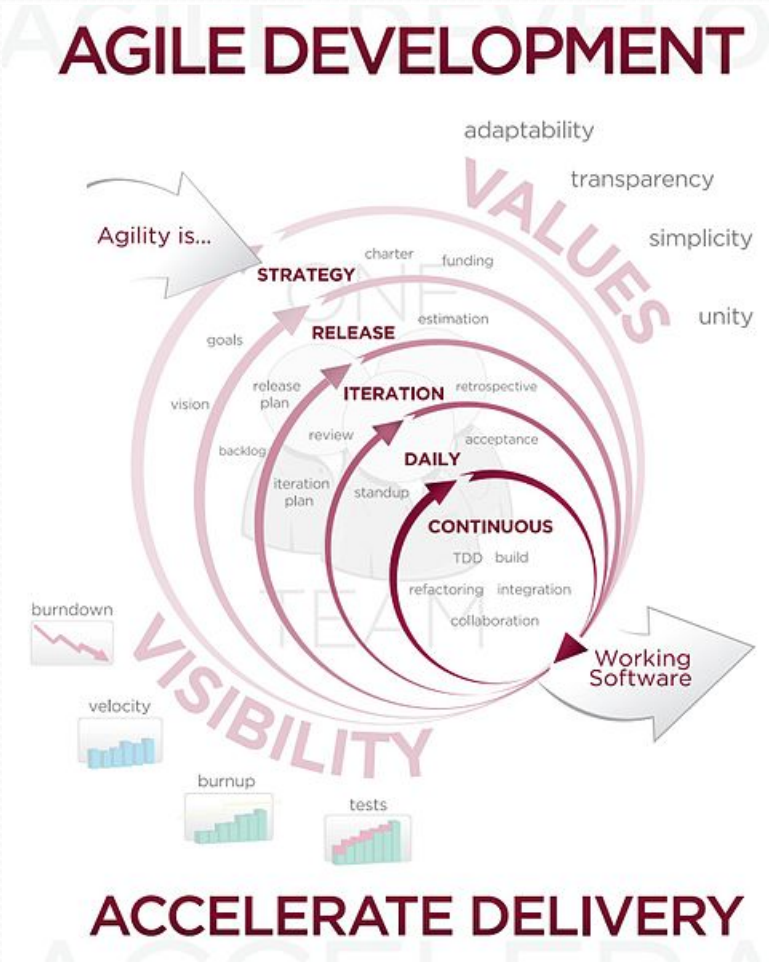
- Software Development History:
- During the 1970s, it was discovered that most large software development projects failed.
- During the 1980s, many of the reasons for those failures began to be recognized.
- In the 1990s, experiments and measurements were used to validate individual methods to prevent failure.
- The current decade is characterized by complete processes to improve success.

Project Failure – the trigger for Agility

- One of the primary causes of project failure was the extended period of time it took to develop a system.
- Costs escalated and requirements changed.

- Agile methods intend to develop systems more quickly with limited time spent on analysis and design.

Agile Development



What is an Agile method? (1)

- Focus on the code rather than the design.
- Based on an iterative approach to software development.
- Intended to deliver working software quickly.
- Evolve quickly to meet changing requirements.
- There are claims that agile methods are probably best suited to small/medium-sized business systems or PC products.

What is an agile method? (2)

- Agile proponents believe:
- Current software development processes are too heavyweight or cumbersome
- Too many things are done that are not directly related to software product being produced
- Current software development is too rigid
- Difficulty with incomplete or changing requirements
- Short development cycles (Internet applications)
- More active customer involvement needed

What is an agile method? (3)

- Agile methods are considered
- Lightweight
- People-based rather than Plan-based
- Several agile methods
- No single agile method
- Extreme Programming (XP) most popular
- No single definition
- Agile Manifesto closest to a definition
- Set of principles
- Developed by Agile Alliance

Customer involvement	The customer should be closely involved throughout the development process. Their role is provide and prioritise new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognised and exploited. The team should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and design the system so that it can accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process used. Wherever possible, actively work to eliminate complexity from the system.

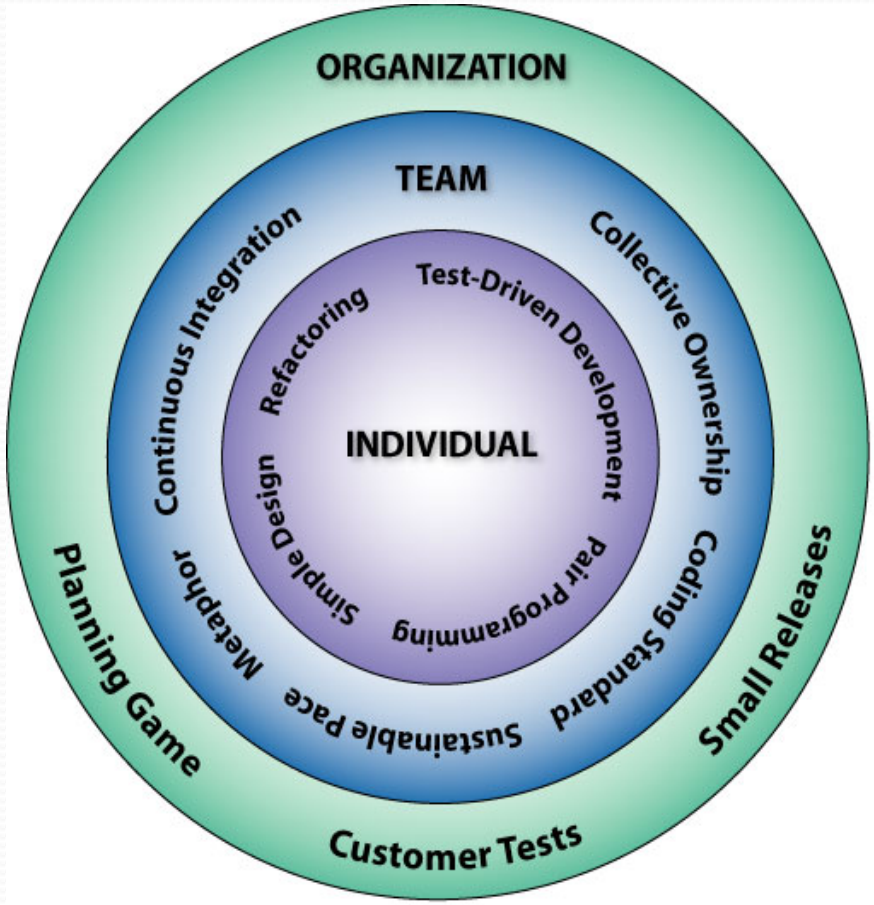
What are the Agile Methodologies?

EXtreme Programming has received the most attention, but here is a list:

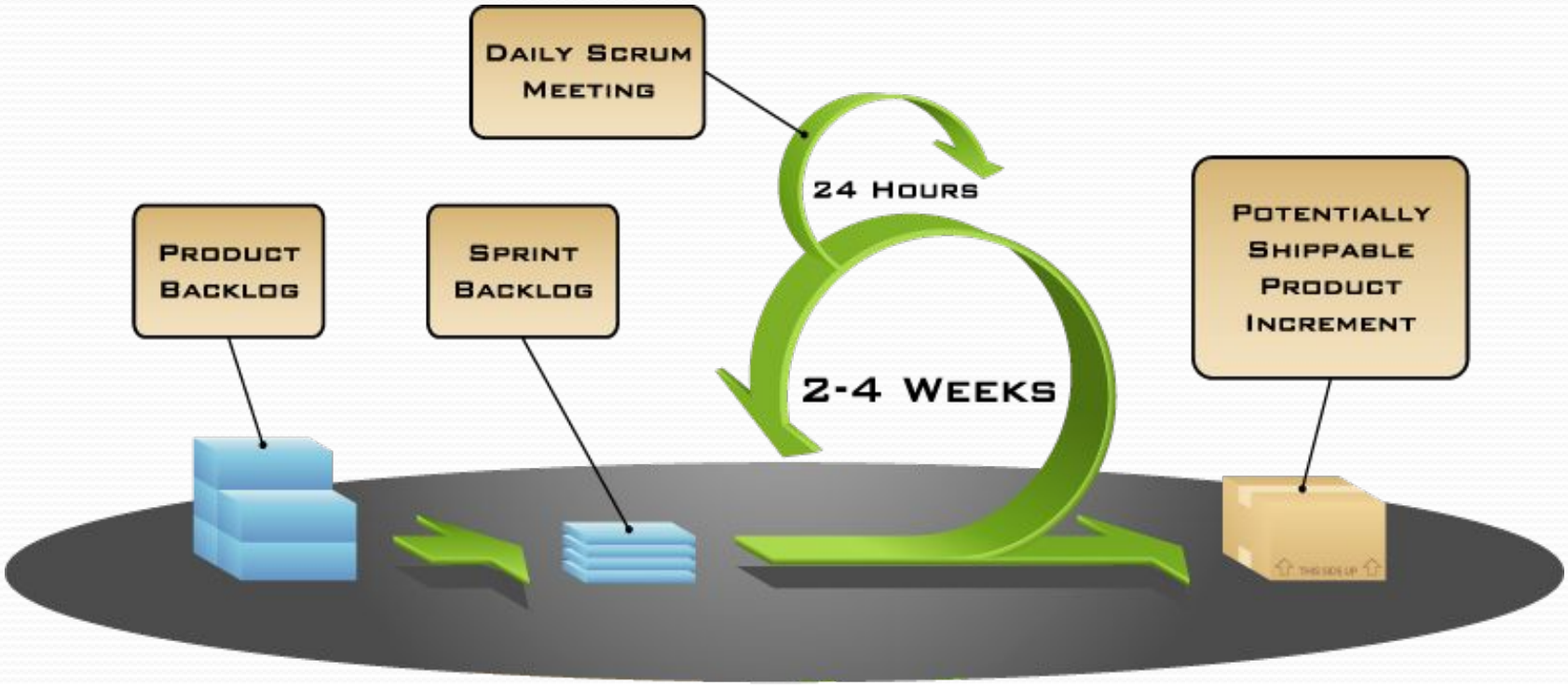
- XP
- SCRUM
- DSDM
- The Crystal Family
- ASD
- FDD
- dX (agile RUP)
- Open Source
- Agile Modeling
- Pragmatic Programming

XP Practices

- The Planning Game
- Small Releases
- Metaphor
- Simple Design
- Test-Driven Development
- Refactoring
- Pair Programming
- Collective Ownership
- Continuous Integration
- 40-Hour Workweek
- On-site Customer
- Coding Standards



How Scrum Works?



Comparison of Different Life Cycle Models

- Iterative waterfall model
- most widely used model.
- But, suitable only for well-understood problems.
- Prototype model is suitable for projects not well understood:
- user requirements
- technical aspects

Comparison of Different Life Cycle Models (CONT.)

- Evolutionary model is suitable for large problems:
- can be decomposed into a set of modules that can be incrementally implemented,
- incremental delivery of the system is acceptable to the customer.
- The spiral model:
- suitable for development of technically challenging software products that are subject to several kinds of risks.

Selection of a Life Cycle Model

- Selection of a model is based on:
- Requirements
- Development team
- Users
- Project type and associated risk

Based On Characteristics Of Requirements

Requirements	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Are requirements easily understandable and defined?	Yes	No	No	No	No	Yes
Do we change requirements quite often?	No	Yes	No	No	Yes	No
Can we define requirements early in the cycle?	Yes	No	Yes	Yes	No	Yes
Requirements are indicating a complex system to be built	No	Yes	Yes	Yes	Yes	No

Based On Status Of Development Team

Development team	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Less experience on similar projects?	No	Yes	No	No	Yes	No
Less domain knowledge (new to the technology)	Yes	No	Yes	Yes	Yes	No
Less experience on tools to be used	Yes	No	No	No	Yes	No
Availability of training if required	No	No	Yes	Yes	No	Yes

Based On User's Participation

Involvement of Users	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
User involvement in all phases	No	Yes	No	No	No	Yes
Limited user participation	Yes	No	Yes	Yes	Yes	No
User have no previous experience of participation in similar projects	No	Yes	Yes	Yes	Yes	No
Users are experts of problem domain	No	Yes	Yes	Yes	No	Yes

Based On Type Of project With Associated Risk

Project type and risk	Waterfall	Prototype	Iterative enhancement	Evolutionary development	Spiral	RAD
Project is the enhancement of the existing system	yes	No	Yes	Yes	No	Yes
Funding is stable for the project	Yes	Yes	No	No	No	Yes
High reliability requirements	No	No	Yes	Yes	Yes	No
Tight project schedule	No	Yes	Yes	Yes	Yes	Yes
Use of reusable components	No	Yes	No	No	Yes	Yes
Are resources (time, money, people etc.) scare?	No	Yes	No	No	Yes	No

Multiple Choice Questions

Note: Select most appropriate answer of the following questions:

Spiral Model was developed by

(a) Bev Littlewood
(b) Berry Boehm
(c) Roger Pressman
(d) Victor Basili

Which model is most popular for student's small projects?

(a) Waterfall model
(b) Spiral model
(c) Quick and fix model
(d) Prototyping model

Which is not a software life cycle model?

(a) Waterfall model
(b) Spiral model
(c) Prototyping model
(d) Capability maturity model

Project risk factor is considered in

(a) Waterfall model
(b) Prototyping model
(c) Spiral model
(d) Iterative enhancement model

SDLC stands for

(a) Software design life cycle
(b) Software development life cycle
(c) System development life cycle
(d) System design life cycle

Multiple Choice Questions

Note: Select most appropriate answer of the following questions:

Build and fix model has

(a) 3 phases
(b) 1 phase
(c) 2 phases
(d) 4 phases

SRS stands for

(a) Software requirements specification
(b) Software requirements solution
(c) System requirements specification
(d) none of the above

Waterfall model is not suitable for

(a) small projects
(b) accommodating change
(c) complex projects
(d) none of the above

RAD stands for

(a) Rapid application development
(b) Relative application development
(c) Ready application development
(d) Repeated application development

10. RAD model was proposed by

(a) Lucent Technologies
(b) Motorola
(c) IBM
(d) Microsoft

Multiple Choice Questions

Note: Select most appropriate answer of the following questions:

If requirements are easily understandable and defined,which model is best suited?

(a) Waterfall model (b) Prototyping model

(c) Spiral model (d) None of the above

If requirements are frequently changing, which model is to be selected?

(a) Waterfall model (b) Prototyping model

(c) RAD model (d) Iterative enhancement model

If user participation is available, which model is to be chosen?

(a) Waterfall model (b) Iterative enhancement model

(c) Spiral model (d) RAD model

If limited user participation is available, which model is to be selected?

(a) Waterfall model (b) Spiral model

(c) Iterative enhancement model (d) any of the above

If project is the enhancement of existing system, which model is best suited?

(a) Waterfall model (b) Prototyping model

(c) Iterative enhancement model (d) Spiral model

Multiple Choice Questions

Note: Select most appropriate answer of the following questions:

Which one is the most important feature of spiral model?

(a) Quality management (b) Risk management

(c) Performance management (d) Efficiency management

Most suitable model for new technology that is not well understood is:

(a) Waterfall model (b) RAD model

(c) Iterative enhancement model (d) Evolutionary development model

Statistically, the maximum percentage of errors belong to the following phase of SDLC

(a) Coding (b) Design

(c) Specifications (d) Installation and maintenance

Which phase is not available in software life cycle?

(a) Coding (b) Testing

(c) Maintenance (d) Abstraction

The development is supposed to proceed linearly through the phase in

(a) Spiral model (b) Waterfall model

(c) Prototyping model (d) None of the above