IS2001: Software Engineering

2. Software Development Process Models -1

Piyumi Kariyawasam

Learning Objectives

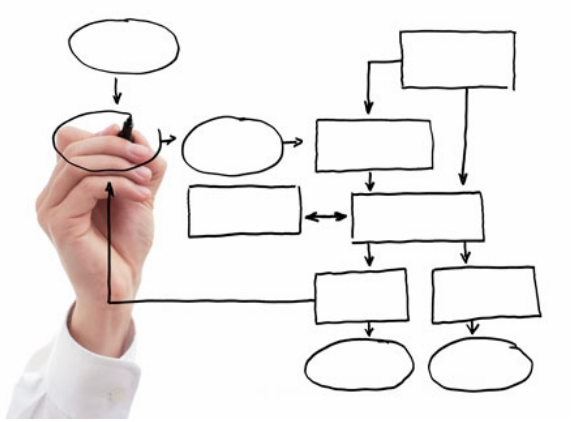
- Describe different process models used for software development
- Identify the most appropriate software process model for a given problem
- Identify how CASE tools can be used to support software process activities

Software Process

- It is important to go through a series of steps to produce high quality software. These steps or the road map followed is called **the software process.**
- Software process is a set of ordered tasks involving activities, constraints and resources that produce a software system
- **Systems Development Life Cycle (SDLC)** represents a life cycle process of a software product.

Need for a Software Process

- A process is important because it imposes consistency and structure on a set of activities
- It guides our actions by allowing us to examine, understand, control and improve the activities of it.



Generic activities in all software processes

- Specification
  - what the system should do and its development constraints
- Design & Development
  - Defining the system design and production of the software system
- Validation
  - checking that the software is what the customer wants
- Evolution
  - Changing/maintaining the software in response to changing demands

Plan-Driven and Agile Processes

- **Plan-driven (traditional)** processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- In **Agile** processes, planning is incremental and it is easier to change the process to reflect changing requirements.
- In practice, most practical processes include elements of both plan-driven and agile approaches.
- There is **no right or wrong** software process. They can be adjusted according to context requirements.

Software Process Models

you need to model the process because:

- when a team writes down a description of its development process it forms a common understanding of the activities, resources and constraints involved in software development
- creating a process model helps the team find inconsistencies, redundancies and commissions in the process, as these problems are noted and corrected the process becomes more effective

Software Process Models

you need to model the process because:

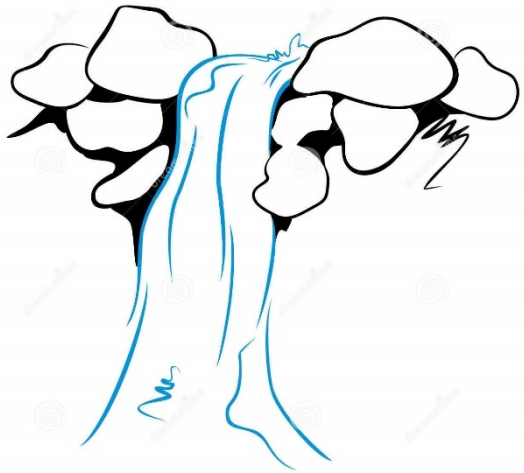
- the model reflects the goals of development and shows explicitly how the product characteristics are to be achieved
- each development is different and a process has to be tailored for different situations, the model helps people to understand these differences

Software Process Models

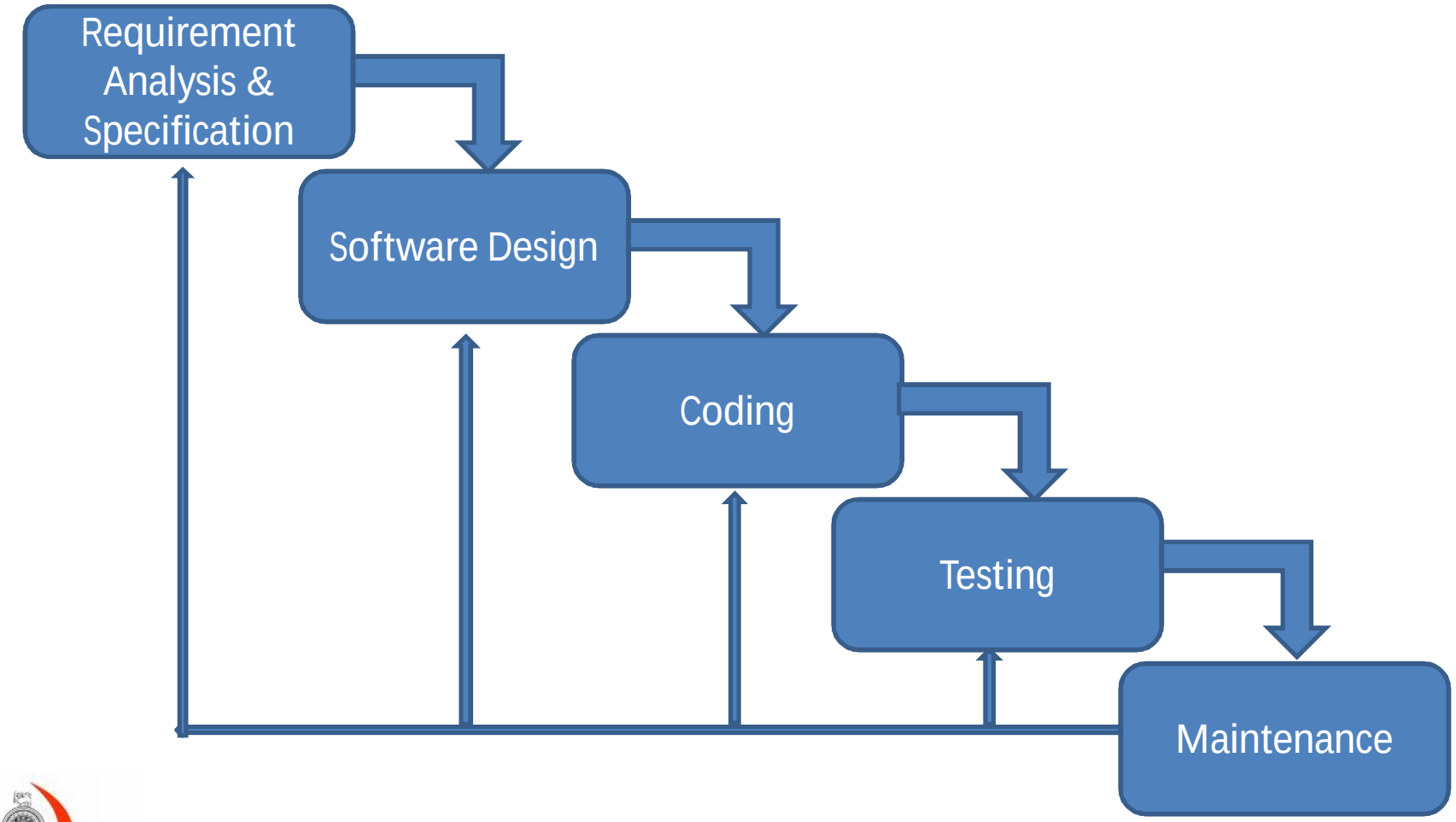
- Waterfall model
- Prototyping models
- The Spiral model
- Incremental development
- Rapid Application development
- Unified Process
- Agile development (A group of models)

Waterfall Model

- Separate and distinct phases of specification and development
- A linear sequential model



Waterfall Model Phases



1. Requirement Analysis & Specification

- The system's services, constraints and goals are established with the consultation with the users.
- This would include the understanding of the information domain for the software, functionality, behavior, performance, interface, security and exceptional requirements.
- This requirements are then specified in a manner which is understandable by both users and the development staff.

2. Software design

- The design process translates requirements into a representation of the software that can be implemented using software tools.
- The major objectives of the design process are the identification of the software components, the software architecture, interfaces, data structures and algorithms.

3. Coding (implementation)

- The design must be translated to a machine readable form.
- During this stage the software design is realized as a set of programs or program units.
- Programming languages or CASE tools can be used to develop software.

4. Testing

- The testing process must ensure that the system works correctly and satisfies the requirements specified.
- After testing, the software system is delivered to the customer.

5. Maintenance

- Software will undoubtedly undergo changes after it is delivered to the customer.
- Errors in the system should corrected and the system should be modified and updated to suit new user requirements.

Advantages of the Waterfall Model

- Easy to understand and implement.
- Widely used and known *(in theory!)*
- Fits other engineering process models: civil, mechanical etc.
- Reinforces good habits: define-before-design, design-before-code
- Identifies deliverables and milestones
- Document driven: *People leave, documents don't*
- Works well on large/mature products and weak teams

Disadvantages of the Waterfall Model

- Doesn't reflect iterative nature of exploratory development.
- Sometimes unrealistic to expect accurate requirements early in a project
- Software is delivered late, delays discovery of serious errors.
- No inherent risk management
- Difficult and expensive to change decisions, "swimming upstream".
- Significant administrative overhead, costly for small teams and projects.

Disadvantages of the Waterfall Model

- The difficulty of accommodating change after the process is underway. One phase has to be complete before moving onto the next phase.
- Only appropriate when the requirements are well-understood and changes will be fairly limited during the design process. Thus the Waterfall model is suitable for projects which have **clear** and **stable requirements**.
- Few business systems have stable requirements.
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.

Evolutionary Development

- Evolutionary development approach is typically used to develop and implement software in a evolutionary manner.
- This approach has been described by Steve McConnell as the "best practice for software development and implementation".

Evolutionary Development

- Early versions of the system are presented to the customer and the system is refined and enhanced based on customer feedback.
- The cycle continues until development time runs out (schedule constraint) or funding for development runs out (resource constraint).

Prototyping

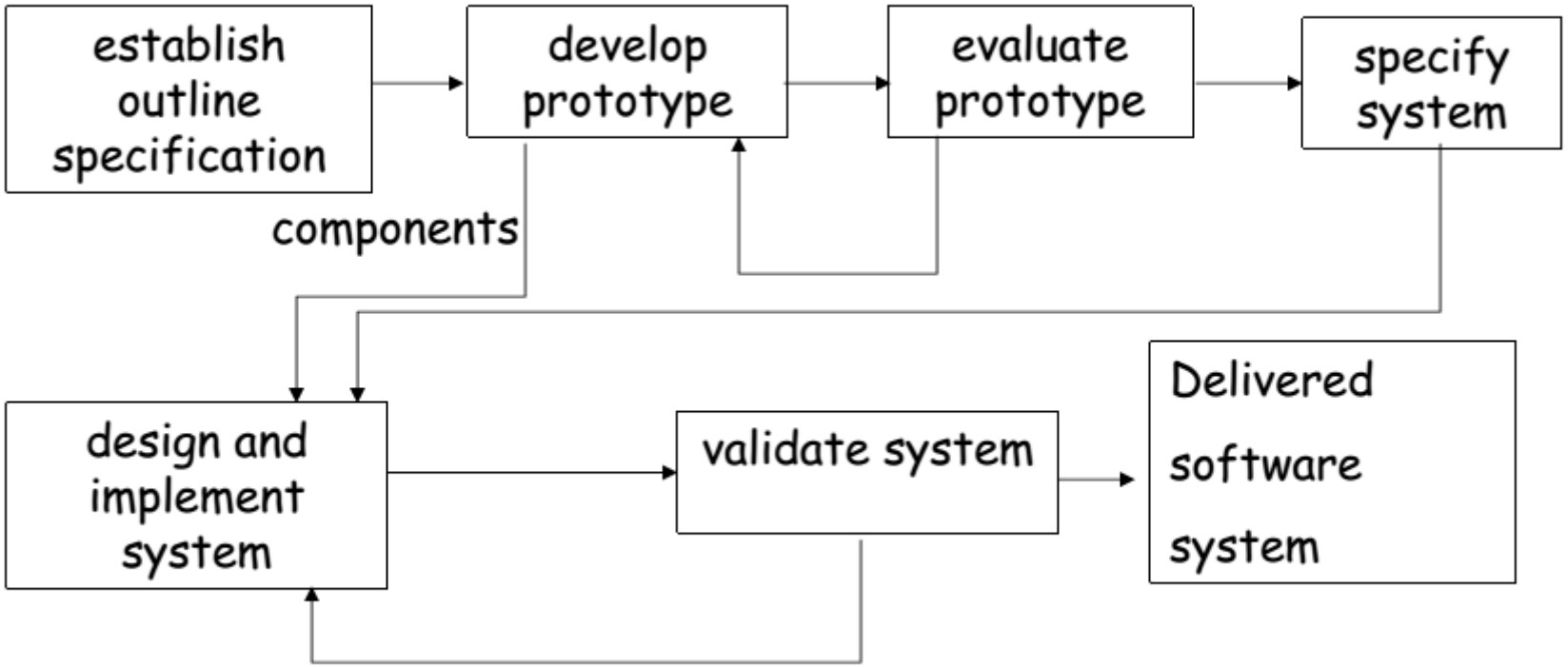
- It is very difficult for end-users to anticipate how they will use new software systems to support their work. If the system is large and complex, it is probably impossible to make this assessment before the system is built and put into use.

- A prototype (a small version of the system) can be used to clear the vague requirements. A prototype should be evaluated with the user participation.

Prototyping

- A prototype is a working model that is functionally equivalent to a component of the product.
- There are two types of Prototyping techniques
  - Throw-away Prototyping
  - Evolutionary Prototyping

Throw-away Prototyping



Throw-away Prototyping

- The objective is to understand the system requirements clearly.
- Starts with poorly understood requirements. Once the requirements are cleared, the system will be developed from the beginning.
- This model is suitable if the requirements are **vague** but **stable**.

Advantages of Throw-away Prototyping

- Reduces risk of incorrect user requirements
- Regular visible progress aids management
- Supports early product marketing

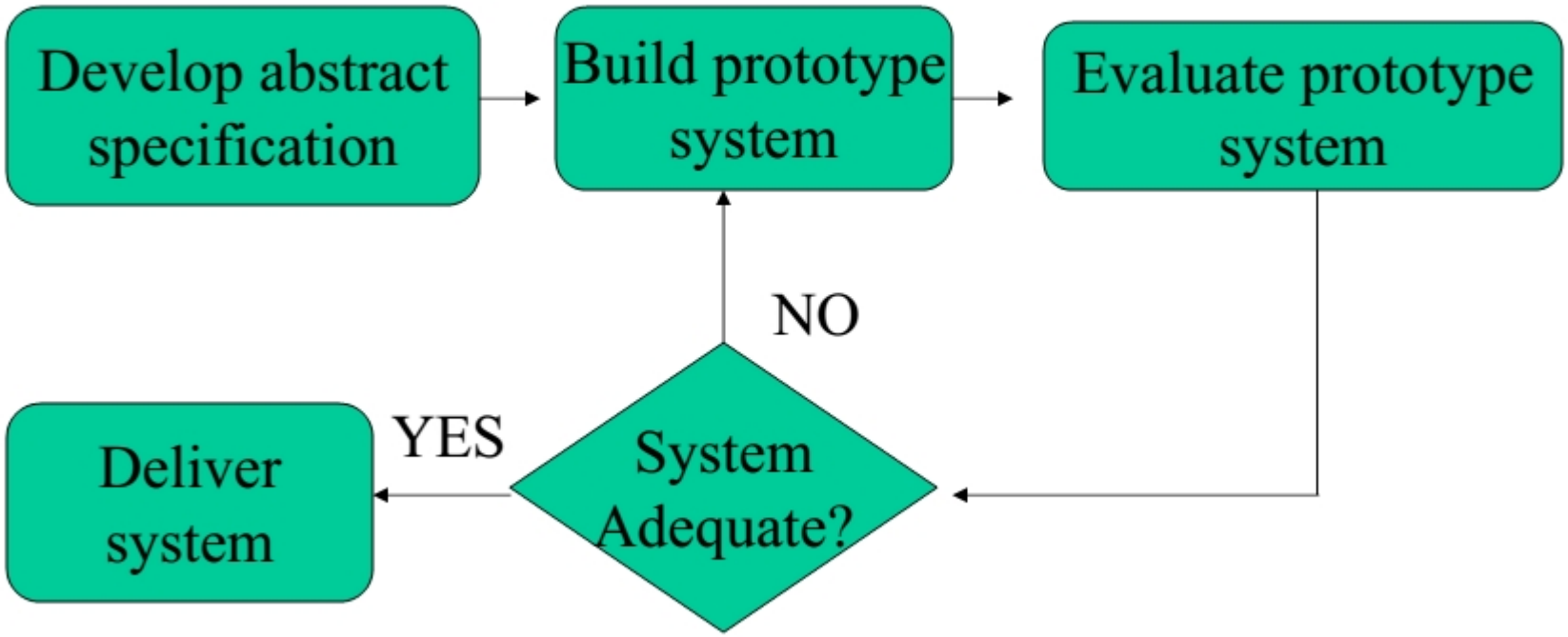
Some problems with Throw-away Prototyping

1. Important features may have been left out of the prototype to simplify rapid implementation. In fact, it may not be possible to prototype some of the most important parts of the system such as safety-critical functions.

2. An implementation has no legal standing as a contract between customer and contractor.

3. Non-functional requirements such as those concerning reliability, robustness and safety cannot be adequately tested in a prototype implementation.

Evolutionary Prototyping



Evolutionary Prototyping

- Advantages
  - Effort of prototype is not wasted
  - Faster than the Waterfall model
  - High level of user involvement from the start
  - Technical or other problems discovered early – risk reduced
  - A working system is available early in the process
  - Misunderstandings between software users and developers are exposed
  - Mainly suitable for projects with **vague** and **unstable requirements**

Evolutionary Prototyping

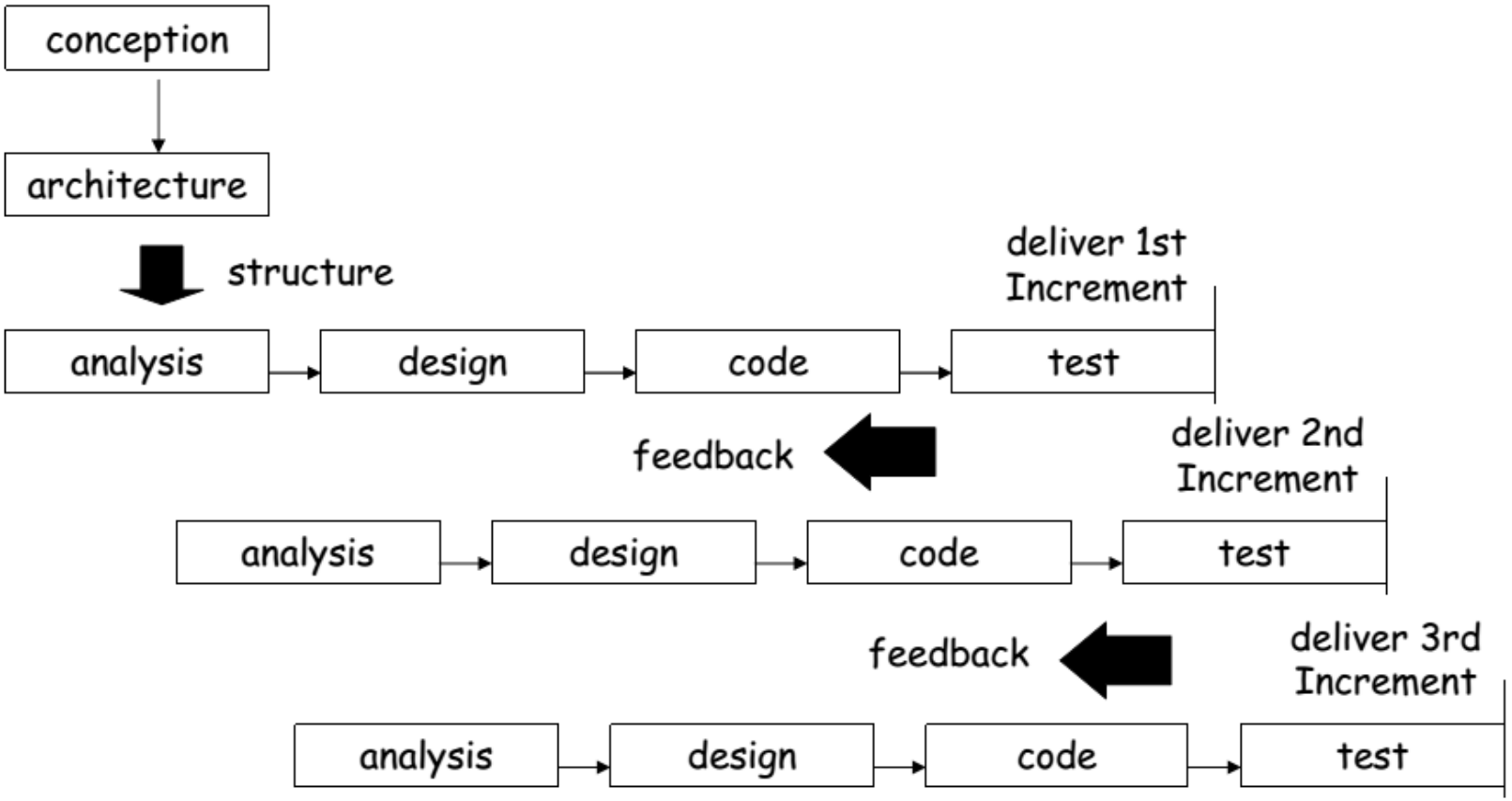
- Disadvantages
  - Prototype usually evolve so quickly that it is not cost-effective to produce great deal of documentation
  - Continual change tends to corrupt the structure of the prototype system. Maintenance is therefore likely to be difficult and costly
  - It is not clear how the range of skills which is normal in software engineering teams can be used effectively for this mode of development
  - Languages which are good for prototyping not always best for final product

- What is the main difference between Throw-away and evolutionary prototypes?

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 Model



Incremental Development

- An overall system architecture is established early in the process to act as a framework.
- Incremental development is more manageable than evolutionary prototyping as the normal software process standards are followed.
- Plans and documentation must be produced.

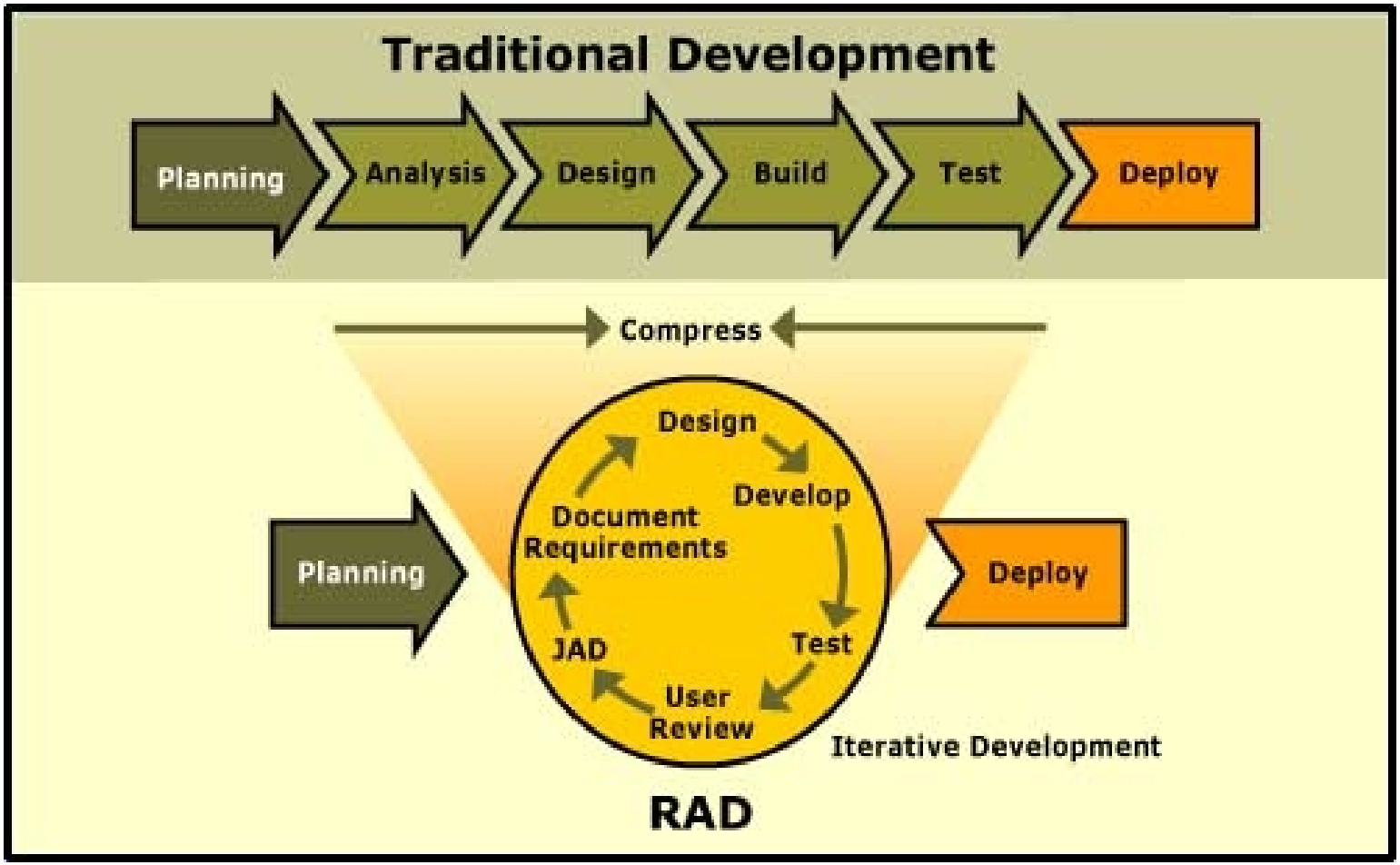
Incremental Development Advantages

- 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.

Rapid Application Development

- Rapid Application Development (RAD) is an incremental software development process model that emphasizes an extremely short development cycle.
- If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a 'fully functional system' within very short time periods (e.g. 60 to 90 days)

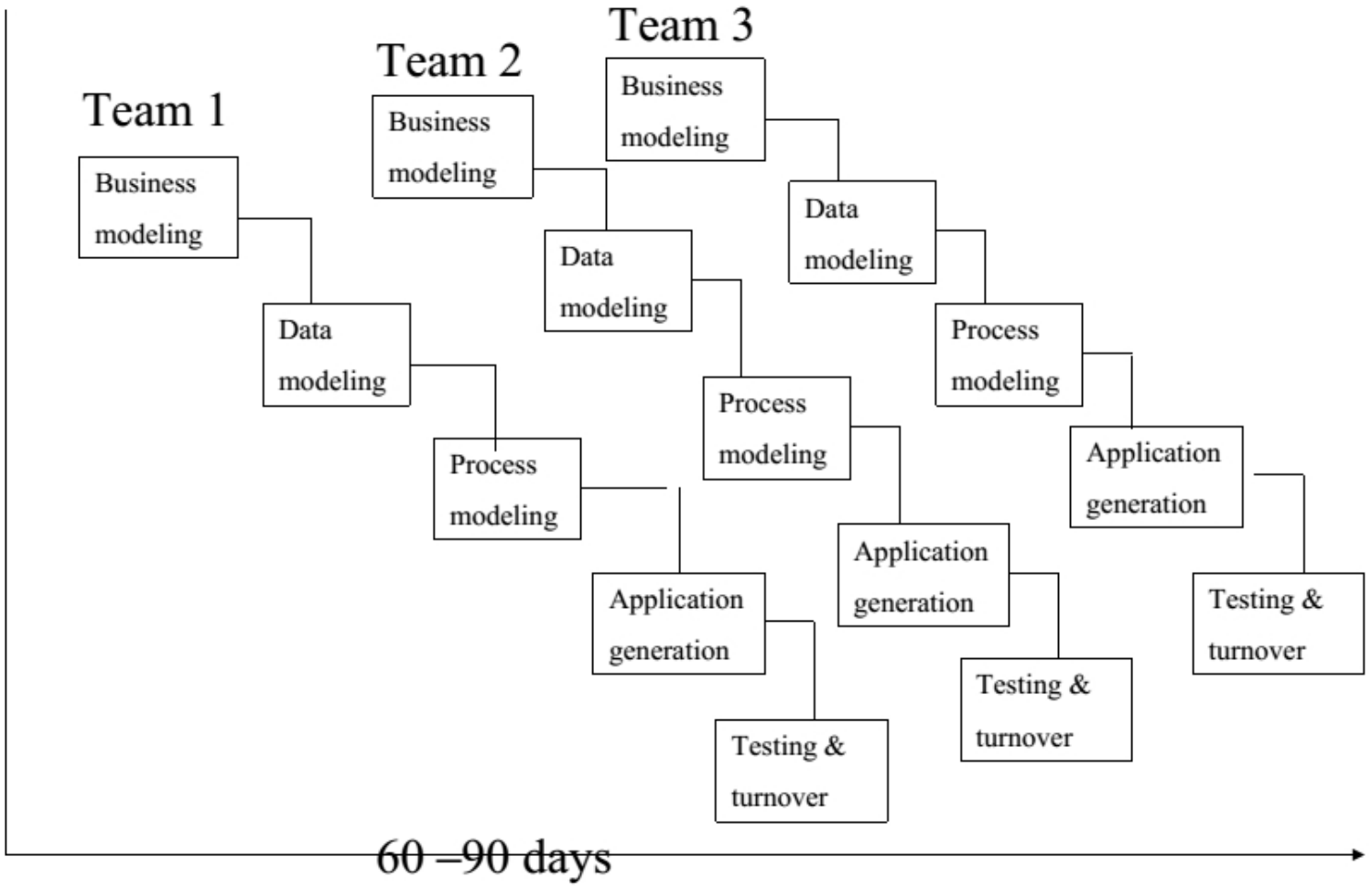
The RAD Model



Types of RAD

- Efficient Development
  balances economy, schedule, and quality
  - Schedule
    - faster than average
  - Economy
    - costs less than average
  - Product
    - better than average quality
- Sensible RAD
  tilts away from economy and quality toward fastest schedule
  - Schedule
    - much faster than average
  - Economy
    - costs a little less than average
  - Product
    - a little better than average quality
- All-out RAD
  "code like hell"
  - Schedule
    - fastest possible
  - Economy
    - costs more than average
  - Product
    - worse than average quality

The RAD Model



Processes in the RAD Model

- **Business modeling**
  - The information flow in a business system considering its functionality.
- **Data Modeling**
  - The information flow defined as part of the business modeling phase is refined into a set of data objects that are needed to support the business
- **Process Modeling**
  - The data objects defined in the data modeling phase are transformed to achieve the information flow necessary to implement business functions.

Processes in the RAD Model

- **Application generation**
  - RAD assumes the use of 4GL or visual tools to generate the system using reusable components.
- **Testing and turnover**
  - New components must be tested and all interfaces must be fully exercised

Stages of RAD: Requirements Planning

- Takes one to four weeks to complete
- is defined during a Joint Requirement Planning (JRP) meeting
- consists of a review of the areas immediately associated with the proposed system
  - produces a broad definition of the system requirements in terms of the functions the system will support
- deliverables from this stage include
  - an outline system area model (entity and process model)
  - a definition of the system's scope
  - a cost justification for the new system

Stages of RAD: User Design

- is defined during Join Application Design (JAD) meetings
- consists of a detailed analysis of the business activities related to the proposed system to outline the design
  - The team defines entity types and creates action diagrams defining the interactions between processes and data
  - System procedures are designed and preliminary layouts of screens are developed
  - Prototypes of critical procedures are built
  - A plan for implementing the system is prepared
- Together with the next stage, User Design consists of a series of iterations

Stages of RAD: Construction

- Developers, working directly with users, finalize the design, build and test the prototype
- The deliverables include documentation and instructions necessary to operate the new application and procedures needed to put the system into operation
- Timebox and parallel development: involves monitoring progresses to complete each task quickly
- The prototype is reviewed by users
  - Requirements document can be modified, so another iteration starts

Stages of RAD: Transition

- the period during which the newly developed system gradually replaces existing systems
  - User acceptance: end of iterations
  - Developers train users to operate the new application
- objectives
  - Install the system in production operation with minimal disruption of normal business activity
  - Maximize the effectiveness of the system in supporting the intended business activities
  - Identify potential future enhancements

RAD Team

should include both developers and users of the system and each person can play several roles

- **User Coordinator**: appointed by the Sponsor to oversee the project from the user perspective
- **Requirements Planning Team**: high-level users participating in the requirements planning
- **User Design Team**: participates in the design meetings
- **User Review Board**: review the system after the construction and decide if modifications are needed
- **Training Manager**: responsible for training users to work with the new system
- **Project Manager**: oversees the development effort
- **Construction (SWAT) Team**: two to six developers highly trained to work together at high speed. SWAT stands for “Skilled Workers with Advanced Tools”. This team builds the system and also participate in the design meetings
- **Meeting Leader**: organizes and conducts the meetings

Problems of RAD

- Criticized because it lacks a methodology and a designed architecture
- Flexibility of RAD systems decreases as applications grow
- RAD tends to fail when
  - Application must interoperate with existing programs
  - Performance and reliability are critical
  - The system cannot be modularized
  - New technologies are used

Problems of RAD

- RAD requires sufficient human resources to create right number of RAD teams
- RAD requires developers and customers who are committed to the rapid-fire activities necessary to get a system completed in a much condensed time frame.
- If a system cannot be properly modularized, building the components necessary for RAD will be problematic.
- RAD is NOT applicable when technical risks are high. This occurs when a new application makes heavy use of new technology or when the new software requires a high degree of interoperability with existing computer programs.

- When to use RAD?

Spiral Development

- Combines some key aspect of the waterfall model and rapid prototyping methodologies
- It is very difficult for end-users to anticipate how they will use new software systems to support their work.
- A prototype (a small version of the software system) can be used to demonstrate concepts, tryout design options and generally to find out more about the problem and its possible solutions.

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.

The Spiral Model

- The spiral model is divided into four main task regions
  1. Determine goals, alternatives, constraints
  2. Evaluate alternatives and risks
  3. Develop and test
  4. Plan

Spiral Model