Software Processes

Part 2

Software Processes

- A set of activities **[*specifying, designing, implementing and testing software systems*]** and associated results **[*requirements spec., design document etc*]** which lead to the production of a software product.

Objectives

- To introduce software process models
- To describe a number of different process models and when they may be used
- To describe outline process models for requirements engineering, software development, testing and evolution
- To introduce CASE technology to support software process activities

Topics covered

- Software process models
- Process iteration
- Software specification
- Software design and implementation
- Software validation
- Software evolution
- Automated process support

The software process

- A structured set of activities and associated results required to develop a software system
  - Specification
  - Design
  - Validation
  - Evolution
- Often companies will have an informal process or a mixed bag of different processes. There is a strong case to argue for process standardization (better communications, less training, process support improvement)
- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective. Each model only provides part of the information about that process.

Software Process models

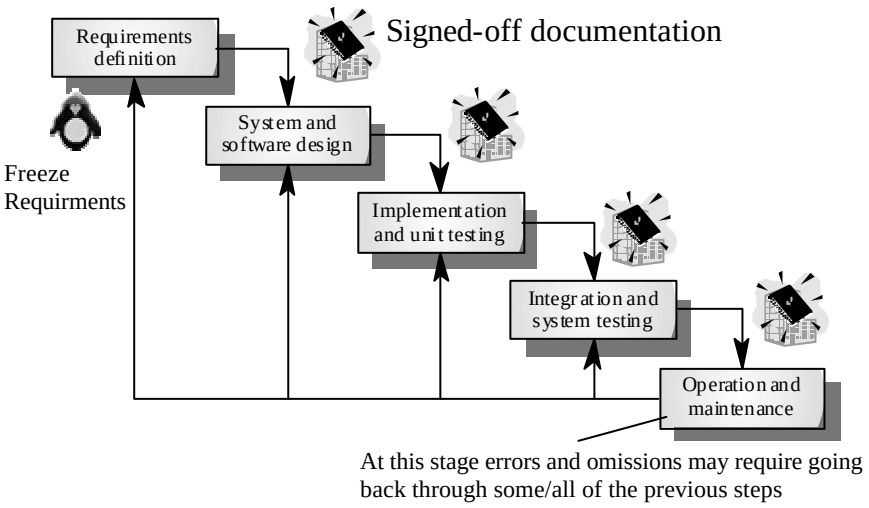
Part 2 - Section 1

Ref : 3.1 Generic software process models

1. The waterfall model
   Separate and distinct phases of specification and development
2. Evolutionary development
   Specification and development are interleaved
3. Formal systems development
   A mathematical system model is formally transformed to an implementation
4. Reuse-based development
   The system is assembled from existing components

Generic Software Process Models(1)

Waterfall model



Waterfall model phases

- Requirements analysis and definition
  - » Used to develop the specification
- System and software design
  - » Here you attempt to fulfill the requirements with H/W or S/W
- Implementation and unit testing
  - » Realize software design as a set of programs
- Integration and system testing
  - » Glue components together and very system meets requirements
- Operation and maintenance
  - » Can be the longest phase where problems are fixed or new functionality is added

This model is a simple management model which seperates design and implementation which can lead to more robust systems which are then more ameanable to change

Waterfall model problems

- The drawback of the waterfall model is the difficulty of accommodating change after the process is underway
- Inflexible partitioning of the project into distinct stages
- This makes it difficult to respond to changing customer requirements which are a fact of life
- Therefore, this model is only appropriate when the requirements are well-understood

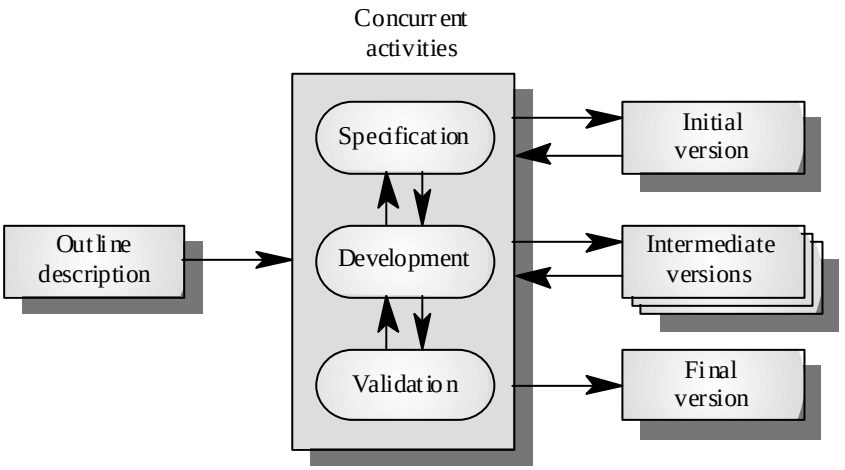
However this model is still extremely popular because of its links to the traditional engineering model, ease of management and tie in with traditional purchasing models!

Generic Software Process Models(2)

Evolutionary development

- **Develop an initial implementation, expose this to the user and then refine the implementation and repeat the process until the system is acceptable.**
- **Two types of evolutionaly developemnt**
  - **Exploratory development *(..and could it do….?)***
    - » **Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements. The system evolves by adding new features as the user proposes them.**
  - **Throw-away prototyping *(…is this what you want…?)***
    - » **Objective is to understand the system requirements. Should start with poorly understood requirements. Consider this as a means to get to grips with what the user requires (may be very apt for User Interfaces)**

Evolutionary development



Evolutionary development

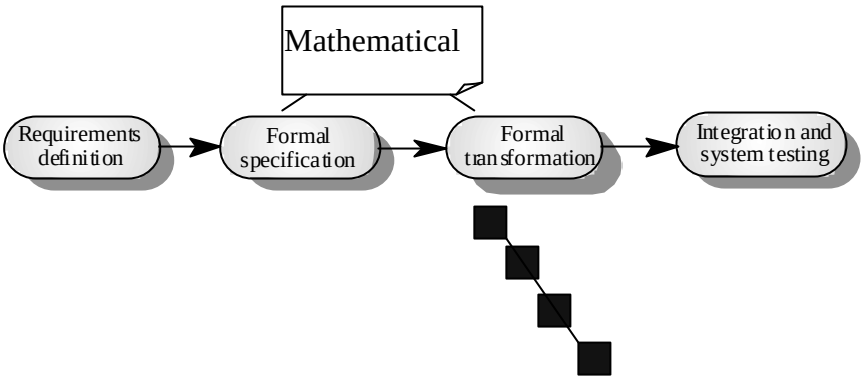
- Good for producing systems that meet the immediate needs of the user
- Problems
  - Lack of process visibility
    - » Producing documentation is vital with any process but it is very difficult to do in the evolutionary development model (constant change). Management may be uncomfortable "running blind"
  - Systems are often poorly structured
    - » Continual change can easily corrupt a system.
  - Special skills (e.g. in languages for rapid prototyping) may berequired
- Applicability
  - For small (<100K LOC) or medium-size (500K LOC) interactive systems Sommerville thinks this is the best approach. Possibly it could be used to pin down requirements and thereafter adopting a more structured approach (e.g. waterfall method)
  - For parts of large systems (e.g. the user interface)
  - For short-lifetime systems

Generic Software Process Models(3)

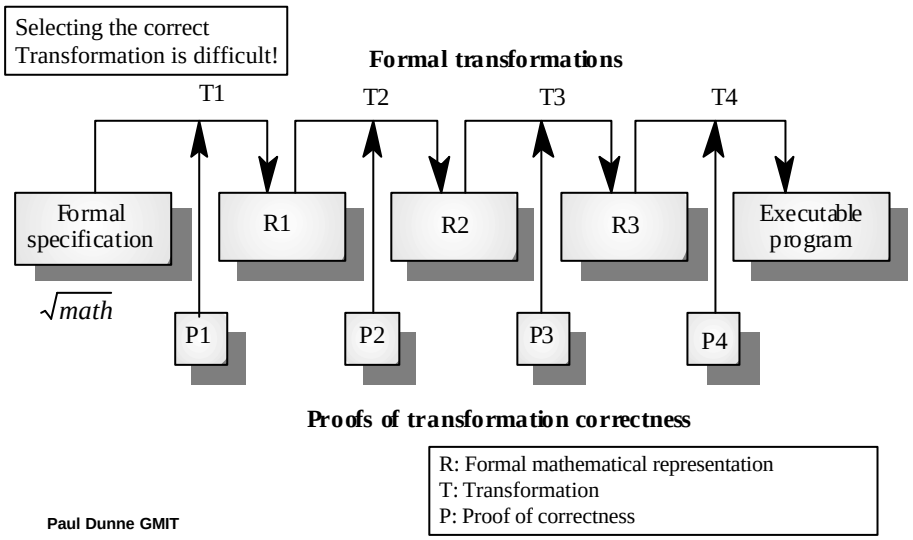
Formal systems development

- Start with a software requirements specification.
- The software requirements specification is refined into a detailed (formal) *mathematical specification* (i.e. expressed in a mathematical notation).
- The *mathematical specification* is passed through a series of transformation (resulting in a number of different representations) to finally appear as an executable program
- Transformations are 'correctness-preserving' so it is straightforward to show that the program conforms to its specification
- Embodied in IBM's 'Cleanroom' approach to software development
- An approach like this, with a number of small steps, is very traceable.

Formal systems development



Formal transformations



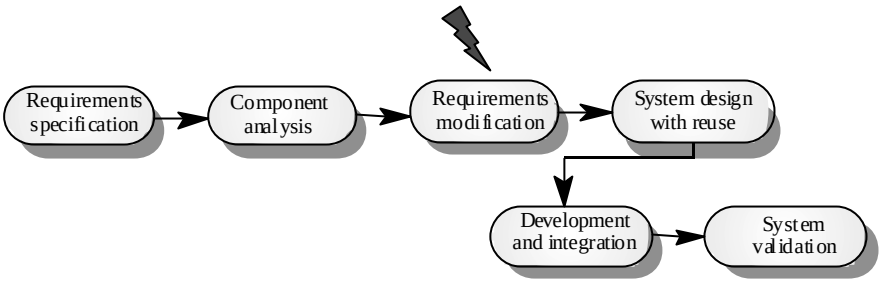
Formal systems development

- **Problems**
  - **Need for specialised skills and training to apply the technique**
  - **Difficult to formally specify some aspects of the system such as the user interface (which is a large part of the development process for most software systems!)**
- **Applicability**
  - **Critical systems especially those where a safety or security case must be made before the system is put into operation**
- **Use**
  - **Outside specialised areas this process is not wide used.**

Generic Software Process Models(4)
Reuse-oriented development

- **Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems**
- **Process stages**
  - **Component analysis**
  - **Requirements modification**
  - **System design with reuse**
  - **Development and integration**
- **This approach is becoming more important but still limited experience with it**

Reuse-oriented development

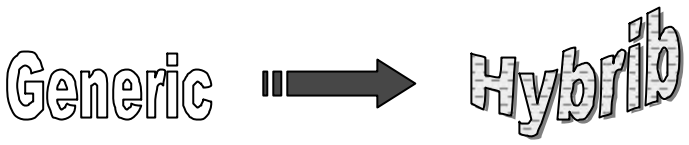


Generic to Hybrib

Section 2

From generic to hybrid

- **Now we will move on from the** generic models **to** hybrid models **which have been developed to address the problems in the real world.**



Process Iteration (ref Chap 3.2)

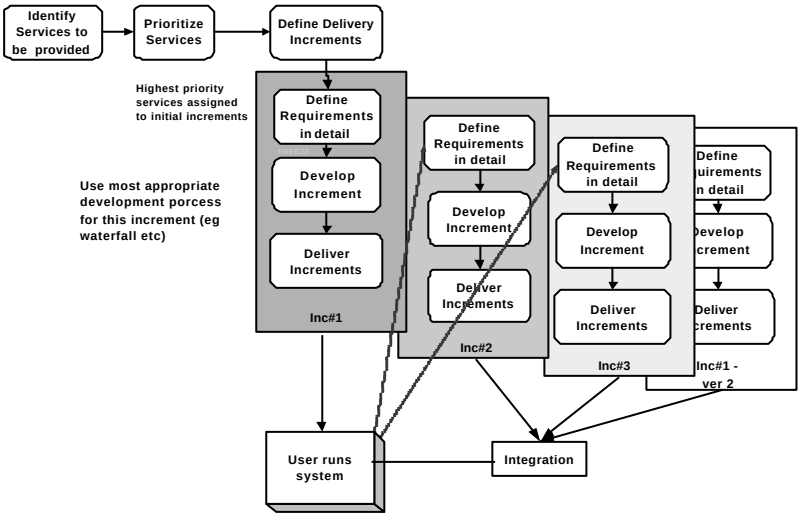
- **System requirements ALWAYS evolve in the course of a project so process iteration where earlier stages are reworked is always part of the process for large systems**
- **Iteration can be applied to any of the generic process models**
- **Two (related) approaches**
  1. **Incremental development**
  2. **Spiral development**
- **The essence of iterative processes is that the specification is developed in conjunction with the software.**
- **Process iteration can cause problems within organizations that are fixed into a traditional procurement model where the specification forms the contract and is required before work begins.**

Approach (1)

Incremental development

- Incremental hopes to combine the best of waterfall process and evolutionary process.
- 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



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 with problems being identified early
  - The highest priority system services tend to receive the most testing
- **Problems**
  - Increments should be small (<20K LOC)
  - Mapping requirements to increments of the right size
  - Common system facilities might not be identified early enough if they are only uncovered incrementally

Incremental development evolution
Extreme programming

- **New approach to development based on the development and delivery of very small increments of functionality**
- **Relies on constant code improvement, user involvement in the development team and pairwise programming**

Approach (2)

Spiral development

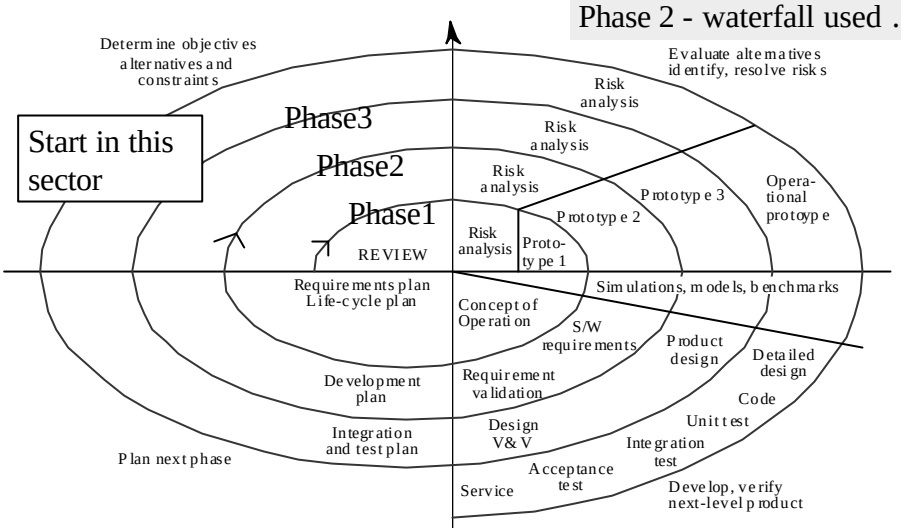
- Originally proposed by Boehm (1988)
- Process is represented as a spiral rather than as a sequence of activities with backtracking
- Each loop in the spiral represents a phase in the process (e.g. The inner could be concerned with system feasibility, next loop from center could be concerned with system requirements definition, the next design etc).
- Where the loops have no name….
  - 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- this is an important distinction with other process models.
- The spiral model encompasses other process models (I.e. Phase 1 could be prototyping etc).

Spiral model sectors

1. **Objective setting**
   - Specific objectives for that phase are defined. Constraints are identified and a management plan is drawn up. Risks are identified and alternative strategies defined As required.
2. **Risk assessment and reduction**
   - Risks are assessed and activities put in place to reduce the key risks (e.g. if there is a risk that the requirements are incorrect a prototype could be built).
3. **Development and validation**
   - A development model for the system is chosen which can be any of the generic models (if safety is critical then development based on formal transformations may be appropriate)
4. **Planning**
   - The project is reviewed and a decision made whether to continue with another loop in the spiral. The next phase of the spiral is planned.

Spiral model of the software process

Phase 1 - prototyping used
Phase 2 - waterfall used …



Sprial Model “walk through”

- **Elaborate objects (performance, functionality etc)**
  1. **System response time <1mSec**
  2. **System to fit in 20cm² space**
- **Enumerate**
  - **Alternative ways of achieving these objectives and Constraints imposed on each of these alternatives**
    - » **1 (a) 2GHz processor --- Min processor size 42cm²**
    - » **1(b) Mutiprocessor computer --- Cost very high**
  - **How does each alternative impact on the other objects**
    - » **1(a) impact on objective #2 ---- RISK #1**
- **Evaluate risks**
  - **Risk #1**
    - » **Do internet search to see if smaller processor exists – success!**
- **Development carried out**
- **Start planning next phase of the process.**

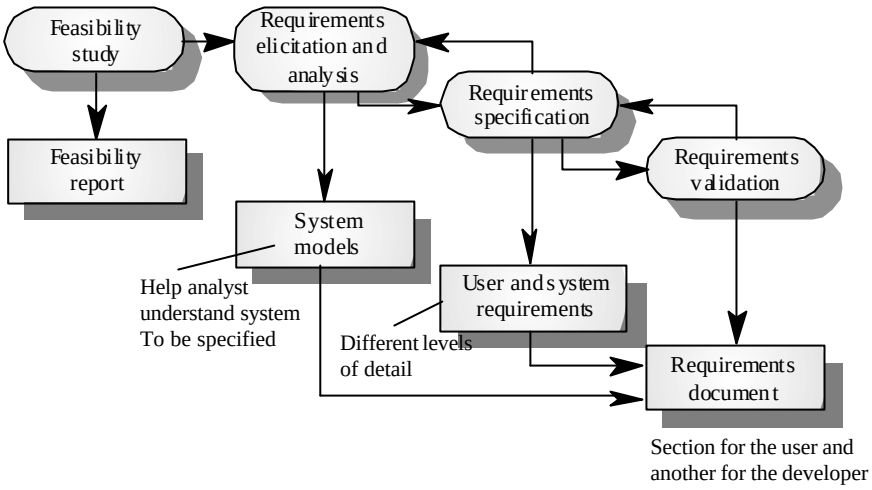
Examining The Basic Process Activities
Specification, Design, Validation and Evolution

Part 2 - Section 3

1 - Software specification

- The process of establishing what services are required and the constraints on the system's operation and development
  - Functional
  - Non-functional requirements
- A critical stage as errors in this stage snowball into the system design and implementation phase.
- Requirements engineering process
  - Feasibility study
  - Requirements elicitation and analysis through observation, discussion, task analysis etc.
  - Requirements specification
    - » This activity transforms information from the the analysis activity into a requirements document.
  - Requirements validation
    - » Are the requirments valid, realistic, consistent etc.

The requirements engineering process

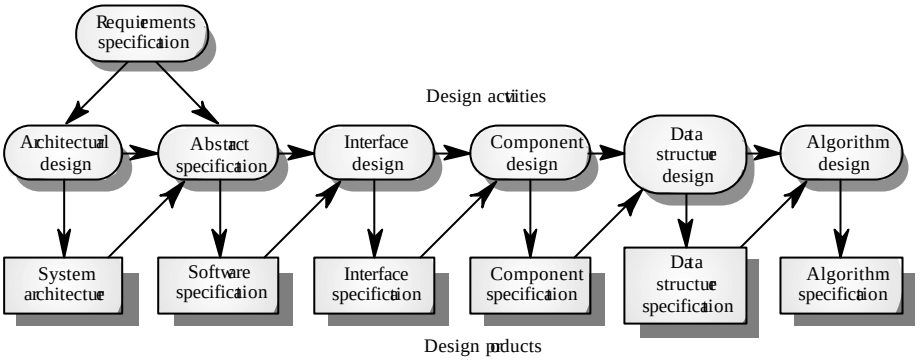


2 - Software design and implementation

- **The process of converting the system specification into an executable system**
- **Always has**
  - **Software design**
    - » **Design a software structure that realises the specification**
  - **Implementation**
    - » **Translate this structure into an executable program**
- **The activities of design and implementation are closely related and may be inter-leaved**

Design

- **The design process involves adding formality and detail as the design is developed. There is constant backtracking to correct earlier designs.**

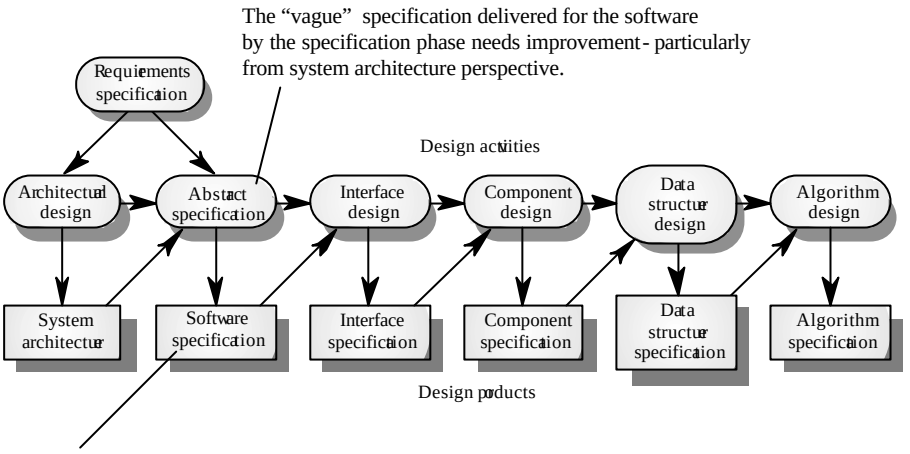


Design process activities

- **Architectural design**
  - **Sub-systems and their relationships identified**
- **Abstract specification**
  - **Sub-system services and constraints identified**
- **Interface design**
  - **Sub-system interfaces to other sub-systems. These interfaces must be unambiguous because it allows the sub-systems to developed and deployed independently.**
- **Component design**
  - **Services are allocated to different components**
- *** Data structure design**
  - **The data structures that will be used are defined and specified.**
- ***Algorithm design**

May be part of the implementation process instead

The software design process



Design methods

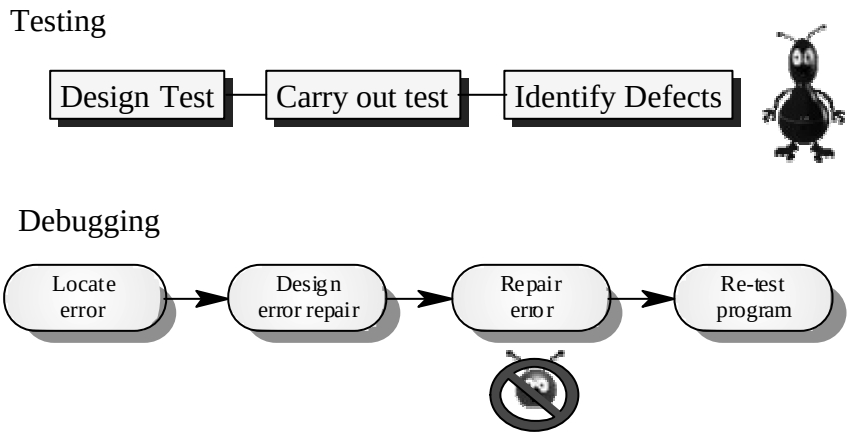
- **Software design is still ad-hoc in many projects**
  - **Natural langauge spec. leads to informal design and then coding commences with the design being modifed as coding proceeds. The original design document has no relationship to the actual system!**
- **Systematic approaches to developing a software design with 'Structured methods' which are sets of notation and guidelines for design.**
  - ***Structured Design, Structured System Analysis, OO Design***
- **The design is usually documented as a set of graphical models with a large amount of associated documentation. CASE tool support is typical.**
- **Structured methods may support some or all of these possible models of a system**

Model	Stuctured method	Notation
Data-flow model Entity-relation-attribute model Structural model Object models	Object-Oriented Design	UML

Implementation
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. Some programmers will start with those components they better understand, develop these before moving onto less well understood components - others will do the opposite.**
- **Programmers carry out some program testing to discover defects in the program and remove these defects in the debugging process**

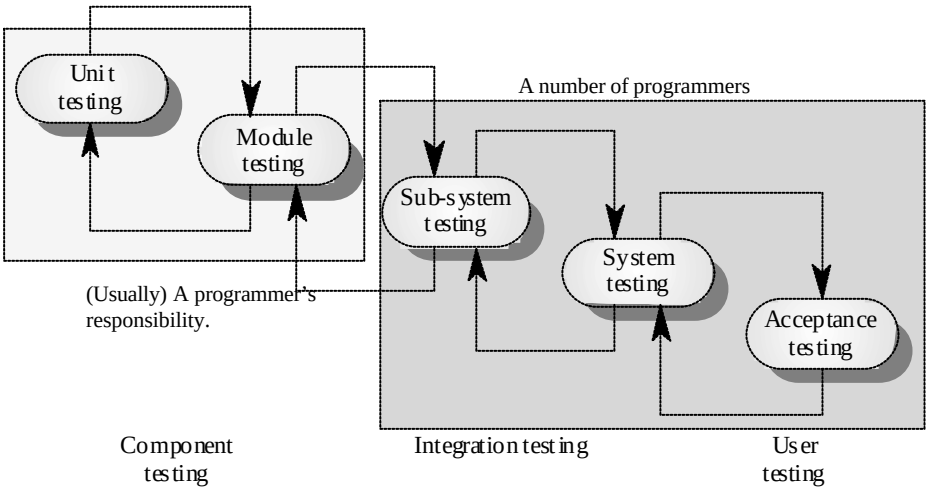
The Testing and Debugging Process



3- Software validation and verification

- Verification and validation (V&V) is intended to show that a system
  - conforms to its specification and
  - meets the requirements of the system customer
- Involves checking processes (such as inspection and reviews) at each stage of the software process as well as system testing on the implemented system.
- Programs should not be tested as a single monolithic unit.
- Testing should occur incrementally in conjunction with system implementation.
- 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

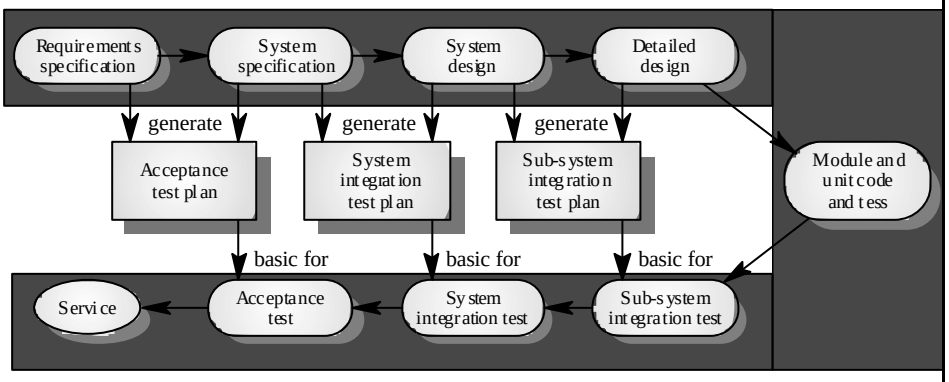
The testing process



Testing stages

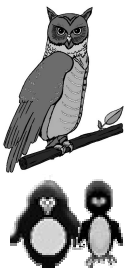
- **Unit testing**
  - **Individual components are tested**
- **Module testing**
  - **Related collections of dependent components are tested**
- **Sub-system testing**
  - **Modules are integrated into sub-systems and tested. The focus here should be on interface testing - the most common problem in the developemnt of large systems is interface mismatch.**
- **System testing**
  - **Testing of the system as a whole. Testing of emergent system properties (ie when the system is built the properties of the system may only then become apparent - a client and a server can be developed independently only when they come together do we have a client-server system)**
- **Acceptance testing (**alpha **- bespoke s/w and** beta**- generic s/w testing)**
  - **Testing with customer data to check that it is acceptable**

Testing phases

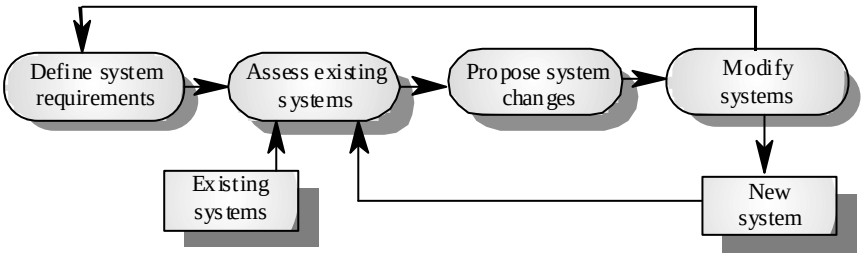


4 - Software evolution

- **Software is inherently flexible and can change therefore it is incorporated into more and more systems (*the flexible employee will get more and more work to do*!)**
- **Change can (relatively easily) be accomadated in software**
- **Historically there has been a demarcation between software development (creative and challenging) and software maintenance (less challenging).**
- **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



Summary

Part 2 - Section 5

Key points

- **Software processes are the activities involved in producing and evolving a software system. They are represented in a software process model**
- **General activities are *specification*, *design* and *implementation*, *validation and evolution***
- **Generic process models describe the organisation of software processes [Waterfall/Evolution/Formal/Reuse]**
- **Hybrid -or Iterative process models describe the software process as a cycle of activities [Incremental/Spiral]**

Key points

- Requirements engineering is the process of developing a software specification
- Design and implementation processes transform the specification to an executable program
- Validation involves checking that the system meets to its specification and user needs
- Evolution is concerned with modifying the system after it is in use
- CASE technology supports software process activities

Questions

3.1 Giving reasons for your answer based on the type of system being developed, suggest the most appropriate generic software process model which might be used as a basic for managing the development of the following systems:

a) **a system to control anti-lock braking in a car;**

b) **A virtual reality system to support software maintenance;**

c) **A college accounting system (replacing existing system)**

d) **An interactive system for railway passengers that finds train times from terminals installed in the stations.**

Questions

3.2 Explain why programs developed using evolutionary development are likely to be difficult to maintain.

3.5 Describe the main activities in the software design process and the outputs of these activities.

3.6 What are the five components of a design method?

3.8 Explain why a software system that is used in a real world environment must change or become progressively less useful.

3.9 Suggest how a CASE technology classification scheme may be helpful to managers responsible for case system procurement.

Model Ans 3.1

(a) Anti-lock braking system
Safety-critical system so method based on formal transformations **with proofs of equivalence between each stage.**

(b) Virtual reality system
System whose requirements cannot be predicted in advance so exploratory programming **model is appropriate.**

(c) College accounting system
System whose requirements should be stable because of existing system therefore waterfall model **is appropriate.**

(d) Interactive timetable
System with a complex user interface but which must be stable and reliable. Should be based on throw-away prototyping **to find requirements then either** incremental development **or** waterfall model**.**

Model Ans 3.6

- **Components of a design method are:**
  - **A defined set of system models**
  - **Rules that apply to these models**
  - **Guidelines for design 'good practice'**
  - **A model of the design process**
  - **Formats for reports on the design**

Model Ans 3.8

- **Systems must change because as they are installed in an environment the environment adapts to them and this adaptation naturally generates new/different system requirements. Furthermore, the system's environment is dynamic and constantly generates new requirements as a consequence of changes to the business, business goals and business policies. Unless the system is adapted to reflect these requirements, its facilities will become out-of-step with the facilities needed to support the business and, hence, it will become less useful.**

Model Ans 3.9

- **A classification scheme can be helpful for system procurement because it helps identify gaps in the CASE tool coverage in an organisation. Procurement may be aimed at filling these gaps. Alternatively, a classification scheme may be used to find tools which support a range of activities - these may represent the most cost effective purchases if funds are limited.**