Coping with change

- **Change is inevitable in all large software projects.**
  - **Business changes lead to new and changed system requirements**
  - **New technologies open up new possibilities for improving implementations**
  - **Changing platforms require application changes**
- **Change leads to rework so the costs of change include both rework (e.g. re-analysing requirements) as well as the costs of implementing new functionality**

Reducing the costs of rework

- **Change avoidance, where the software process includes activities that can anticipate possible changes before significant rework is required.**
  - **For example, a prototype system may be developed to show some key features of the system to customers.**
- **Change tolerance, where the process is designed so that changes can be accommodated at relatively low cost.**
  - **This normally involves some form of incremental development. Proposed changes may be implemented in increments that have not yet been developed. If this is impossible, then only a single increment (a small part of the system) may have been altered to incorporate the change.**

Ways to Cope with change

- **Software Prototyping**

  where a version of the system or part of the system is developed quickly to check the customer's requirements and the feasibility of some design decisions.
- **Incremental delivery**

  where system increments are delivered to the customer for comment and experimentation.

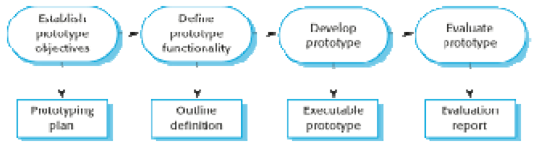
Software prototyping

- **A prototype is an initial version of a system used to demonstrate concepts and try out design options.**
- **A prototype can be used in:**
  - **The requirements engineering process to help with requirements elicitation and validation;**
  - **In design processes to explore options and develop a UI design;**
  - **In the testing process to run back-to-back tests.**

Benefits of prototyping

- Improved system usability.
- A closer match to users' real needs.
- Improved design quality.
- Improved maintainability.
- Reduced development effort.

The process of prototype development



Prototype development

- May be based on rapid prototyping languages or tools
- May involve leaving out functionality
  - Prototype should focus on areas of the product that are not well-understood;
  - Error checking and recovery may not be included in the prototype;
  - Focus on functional rather than non-functional requirements such as reliability and security

Throw-away prototypes

- Developers are sometimes pressured by managers to deliver throwaway prototypes. However, this is usually unwise because:
  1. It may be impossible to tune the system to meet non-functional requirements;
  2. Prototypes are normally undocumented;
  3. The prototype structure is usually degraded through rapid change;
  4. The prototype probably will not meet normal organisational quality standards.

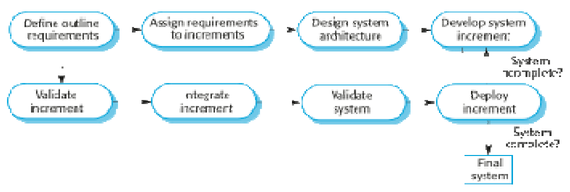
Incremental delivery

- Incremental delivery is an approach to software development where some of the developed increments are delivered to the customer and deployed for use in an operational environment
- 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 and delivery

- Incremental development
  - – Develop the system in increments and evaluate each increment before proceeding to the development of the next increment;
  - – Normal approach used in agile methods;
  - – Evaluation done by user/customer.
- Incremental delivery
  - – Deploy an increment for use by end-users;
  - – More realistic evaluation about practical use of software;
  - – Difficult to implement for replacement systems as increments have less functionality than the system being replaced.

Incremental delivery



Incremental delivery advantages

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

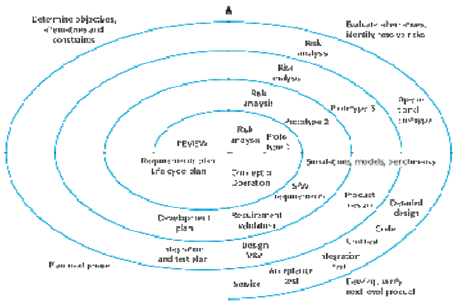
Incremental delivery problems

- Most systems require a set of basic facilities that are used by different parts of the system.
  - As requirements are not defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all increments.
- The essence of iterative processes is that the specification is developed in conjunction with the software.
- Iterative development can also be difficult when a replacement system is being developed. Users want all of the functionality of the old system and are often unwilling to experiment with an incomplete new system Therefore, getting useful customer feedback is difficult.

Boehm's spiral model

- 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.

Boehm's spiral model of the software process



Spiral model sectors

- Objective setting
  - Specific objectives for the phase are identified.
  - Constraints on the process and the product are identified
  - a detailed management plan is drawn up. Project risks are identified.
- Risk assessment and reduction
  - Risks are assessed and activities put in place to reduce the key risks.
  - For each of the identified project risks, a detailed analysis is carried out.
  - Steps are taken to reduce the risk

Spiral model sectors

- Development and validation
  - A development model for the system is chosen which can be any of the generic models.
  - For example, throwaway prototyping may be the best development approach if user interface risks are dominant.
- Planning

  The The project is reviewed and a decision made whether to continue with a further loop of the spiral. If it is decided to continue, plans are drawn up for the next phase of the project.

Spiral model usage

- Spiral model has been very influential in helping people think about iteration in software processes and introducing the risk-driven approach to development.
- In practice, however, the model is rarely used as published for practical software development.