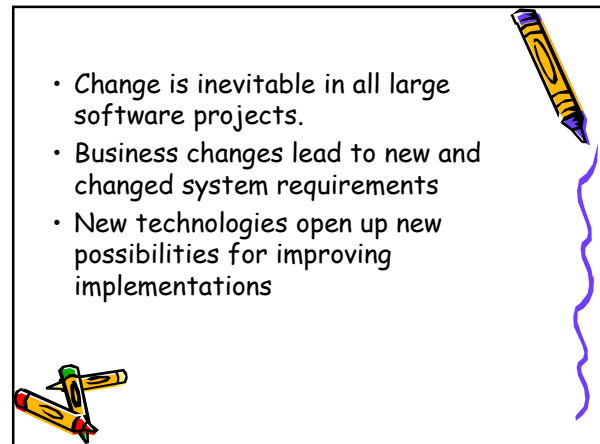
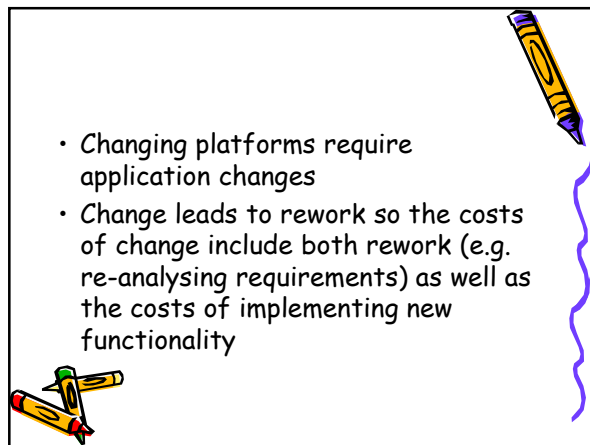


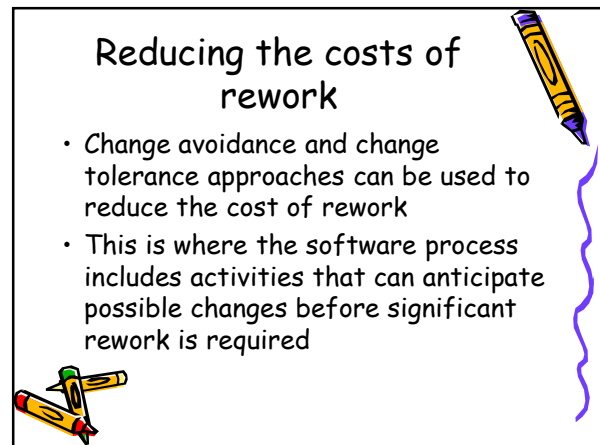
## Coping with change in software engineering



- 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

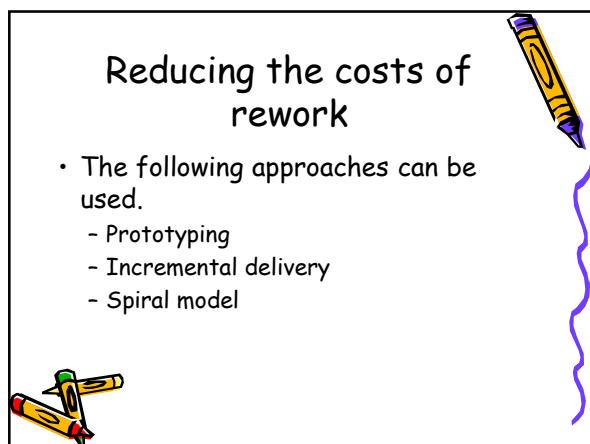


- Changing platforms require application changes
- Change leads to rework so the costs of change include both rework (e.g. re-analysing requirements) as well as the costs of implementing new functionality



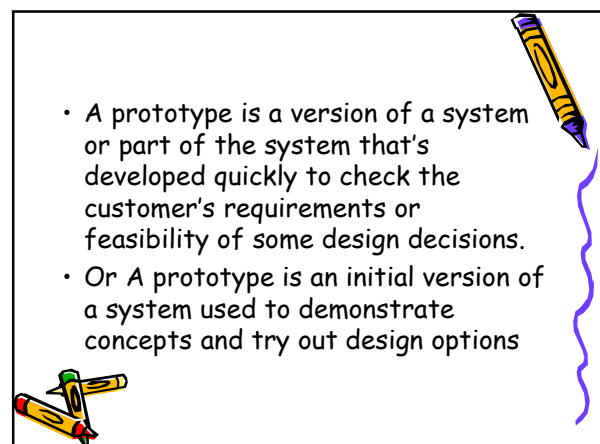
### Reducing the costs of rework

- Change avoidance and change tolerance approaches can be used to reduce the cost of rework
- This is where the software process includes activities that can anticipate possible changes before significant rework is required



### Reducing the costs of rework

- The following approaches can be used.
  - Prototyping
  - Incremental delivery
  - Spiral model



- A prototype is a version of a system or part of the system that's developed quickly to check the customer's requirements or feasibility of some design decisions.
- Or A prototype is an initial version of a system used to demonstrate concepts and try out design options

- a prototype is useful when a customer or developer is not sure of the requirements, or of algorithms, efficiency, business rules, response time, etc.
- In prototyping, the client is involved throughout the development process, which increases the likelihood of client acceptance of the final implementation.

## Purpose of prototyping

- In the **requirements engineering**, a prototype can help with the elicitation and validation of system requirements.
  - It allows the users to experiment with the system, and so, refine the requirements.
  - They may get new ideas for requirements, and find areas of strength and weakness in the software.

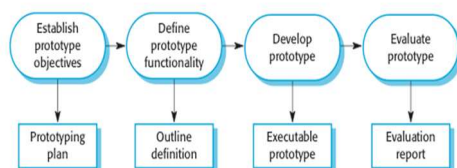
## Purpose of prototyping

- Furthermore, as the prototype is developed, it may reveal errors in the requirements.
- The specification may be then modified to reflect the changes.

## Purpose of prototyping

- In the **system design**, a prototype can help to carry out design experiments to check the feasibility of a proposed design.
  - For example, a database design may be prototype-d and tested to check if it supports efficient data access for the most common user queries.

## The prototype development Model



## The phases of a prototype

- **Establish objectives:**
  - The objectives of the prototype should be made explicit from the start of the process.
  - Is it to validate system requirements, or demonstrate feasibility, etc.

## The phases of a prototype

- **Define prototype functionality:**
  - Decide what are the inputs and the expected output from a prototype.
  - To reduce the prototyping costs and accelerate the delivery schedule, you may ignore some functionality, such as response time and memory utilization unless they are relevant to the objective of the prototype.



## The phases of a prototype

- **Develop the prototype:** The initial prototype is developed that includes only user interfaces.
- **Evaluate the prototype:** Once the users are trained to use the prototype, they then discover requirements errors.



## The phases of a prototype

- Using the feedback both the specifications and the prototype can be improved.
- If changes are introduced, then a repeat of steps 3 and 4 may be needed



## Prototyping

- Prototyping is not a standalone, complete development methodology, but rather an approach to be used in the context of a full methodology (such as incremental, spiral, etc).



## Incremental Model

- Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle.
- Incremental development is done in steps from analysis, design, implementation, testing/verification, maintenance.



## Incremental Model

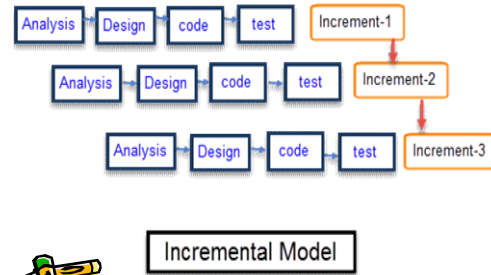
- Each iteration passes through the **requirements, design, coding and testing phases**.
- And each subsequent release of the system adds function to the previous release until all designed functionality has been implemented.



## Incremental Development

- Incremental development is based on the idea of developing an initial implementation, exposing this to user feedback, and evolving it through several versions until an acceptable system has been developed.
- The activities of a process are not separated but interleaved with feedback involved across those activities

## Incremental Development Model



## Incremental Development Model

Incremental Phases	Activities performed in incremental phases
Requirement Analysis	*Requirement and specification of the software are collected
Design	*Some high-end function are designed during this stage
Code	*Coding of software is done during this stage
Test	*Once the system is deployed, it goes through the testing phase

## Incremental Development Model

- Each system increment reflects a piece of the functionality that is needed by the customer.
- Generally, the early increments of the system should include the most important or most urgently required functionality.

## Incremental Development Model

- This means that the customer can evaluate the system at early stage in the development to see if it delivers what's required.
- If not, then only the current increment has to be changed and, possibly, new functionality defined for later increments.

## Characteristics of an Incremental Model

- System development is broken down into many mini development projects
- Partial systems are successively built to produce a final total system
- Highest priority requirement is tackled first
- Once the requirement is developed, requirement for that increment are frozen

### Advantages of Incremental

- The software will be generated quickly during the software life cycle
- It is flexible and less expensive to change requirements and scope
- Throughout the development stages changes can be done
- This model is less costly compared to others
- A customer can respond to each building
- Errors are easy to be identified

### DisAdvantages of Incremental

- Problems might be caused due to system architecture and as such not all requirements can be collected up front for the entire software lifecycle
- Each iteration phase is rigid and does not overlap each other

### When to use Incremental models

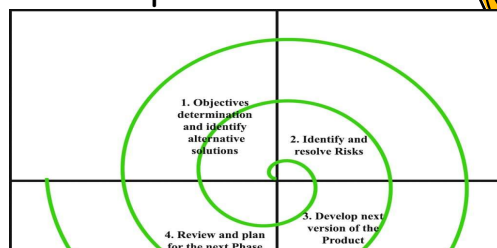
- Requirements of the system are clearly understood
- When demand for an early release of a product arises
- When software engineering team are not very well skilled or trained
- When high-risk features and goals are involved
- Such methodology is more in use for web application and product based companies

### Spiral model

- The spiral model is a risk-driven model where the process is represented as spiral rather than a sequence of activities.
- It was designed to include the best features from the waterfall and prototyping models, and introduces a new component; risk-assessment

- The initial phase of the Spiral model is the early stages of Waterfall Life Cycle that are needed to develop a software product.
- the project is delivered in loops.
- Each loop in the Spiral model is the phases of the software development process.

### Spiral model



## Spiral model

- In this model, the exact number of phases for developing a product varied based on some constraints and by project manager which calculates the project risks.
- Here the project manager dynamically decides the number of phases and hence play a significant role in the development of a product using the spiral model

## Spiral model

- Each loop in the spiral represents a phase.
- Thus the first loop might be concerned with system feasibility, the next loop might be concerned with the requirements definition, the next loop with system design, and so on.

## Phases in the Spiral

- Each loop in the spiral is split into four sectors:
- **Objective setting:**
  - The objectives and risks for that phase of the project are defined.

## Phases in the Spiral

- **Risk assessment and reduction:**
  - For each of the identified project risks, a detailed analysis is conducted, and steps are taken to reduce the risk.
  - For example, if there's a risk that the requirements are inappropriate, a prototype may be developed.

## Phases in the Spiral

- **Development and validation:**
  - After risk evaluation, a process model for the system is chosen.
  - So if the risk is expected in the user interface then we must prototype the user interface.
  - If the risk is in the development process itself then use the waterfall model.

## Phases in the Spiral

- **Planning:**
  - The project is reviewed and a decision is made whether to continue with a further loop or not

### Advantages

- Suitable for large projects: Spiral models are recommended when the project is large, bulky or complex to develop.
- Requirements flexibility: All the specific requirements needed at later stages can be included precisely if the development is done using this model



### Advantages

- Risk Handling: There are a lot of projects that have un-estimated risks involved with them. For such projects, the spiral model is the best SDLC model to pursue because it can analyze risk as well as handling risks at each phase of development.



### Advantages

- Customer Satisfaction: Customers can witness the development of product at every stage and thus, they can let themselves habituated with the system and throw feedbacks accordingly before the final product is made.



### Disadvantages

- **Complex:** The Spiral Model is much more complex than other SDLC models.
- **Expensive:** Spiral Model is not suitable for small projects as it is expensive.
- **Difficulty in time management:** As the number of phases is unknown at the start of the project, so time estimation is very difficult.



### Disadvantages

- **Too much dependable on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced expertise, it is going to be a failure to develop a project using this model.



### Questions

