***Software Engineering***

***Lecture 6***

***Topics covered***

* Coping with change
* Process improvement

***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-analyzing** **requirements**) as well as the costs of implementing new functionality

***Reducing the costs of rework***

* Change anticipation, 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 be altered to incorporate the change.

***Coping with changing requirements***

* System 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 design decisions. This approach supports change anticipation.
* Incremental delivery, where system increments are delivered to the customer for comment and experimentation. This supports both change avoidance and change tolerance.

***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 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***

2.9 PrototypeProcess.eps

***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***

* Prototypes should be discarded after development as they are not a good basis for a production system:
  + It may be impossible to tune the system to meet non-functional requirements;
  + Prototypes are normally undocumented;
  + The prototype structure is usually degraded through rapid change;
  + The prototype probably will not meet normal organizational quality standards.

***Incremental delivery***

* 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 proxy.

***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***

2.10 Incremental-delivery.eps

***Incremental delivery 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 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.
  + However, this conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract.

***Process improvement***

* Many software companies have turned to software process improvement as a way of enhancing the quality of their software, reducing costs or accelerating their development processes.
* Process improvement means understanding existing processes and changing these processes to increase product quality and/or reduce costs and development time.

***Approaches to improvement***

* **The process maturity approach,** which focuses on improving process and project management and introducing good software engineering practice.
  + The level of process maturity reflects the extent to which good technical and management practice has been adopted in organizational software development processes.
* **The agile approach,** which focuses on iterative development and the reduction of overheads in the software process.
  + The primary characteristics of agile methods are rapid delivery of functionality and responsiveness to changing customer requirements.

***The process improvement cycle***

26.3 Process improvement.eps

***Process improvement activities***

***Process measurement: -***

* **You** measure one or more attributes of the software process or product. These measurements form a baseline that helps you decide if process improvements have been effective.

***Process analysis: -***

* **The** current process is assessed, and process weaknesses and bottlenecks are identified. Process models (sometimes called process maps) that describe the process may be developed.

***Process change: -***

* **Process** changes are proposed to address some of the identified process weaknesses. These are introduced and the cycle resumes to collect data about the effectiveness of the changes.

***Process measurement***

* **Wherever possible, quantitative process data should be collected**
  + However, where organisations do not have clearly defined process standards this is very difficult as you don’t know what to measure. A process may have to be defined before any measurement is possible.
* **Process measurements should be used to assess process improvements**
  + But this does not mean that measurements should drive the improvements. The improvement driver should be the organizational objectives.

***Process metrics***

* Time taken for process activities to be completed
  + E.g., Calendar time or effort to complete an activity or process.
* Resources required for processes or activities
  + E.g., Total effort in person-days.
* Number of occurrences of a particular event
  + E.g., Number of defects discovered.

***Capability maturity levels***

26.10 StagesCMMI.eps

***The SEI capability maturity model***

* ***Initial***
  + Essentially uncontrolled
* ***Repeatable***
  + Product management procedures defined and used
* ***Defined***
  + Process management procedures and strategies defined and used
* ***Managed***
  + Quality management strategies defined and used
* ***Optimising***
  + Process improvement strategies defined and used

***Key points***

* Software processes are the activities involved in producing a software system. Software process models are abstract representations of these processes.
* General process models describe the organization of software processes.
  + Examples of these general models include the ‘waterfall’ model, incremental development, and reuse-oriented development.
* Requirements engineering is the process of developing a software specification.
* Design and implementation processes are concerned with transforming a requirements specification into an executable software system.
* Software validation is the process of checking that the system conforms to its specification and that it meets the real needs of the users of the system.
* Software evolution takes place when you change existing software systems to meet new requirements. The software must evolve to remain useful.
* Processes should include activities such as prototyping and incremental delivery to cope with change.
* Processes may be structured for iterative development and delivery so that changes may be made without disrupting the system as a whole.
* The principal approaches to process improvement are agile approaches, geared to reducing process overheads, and maturity-based approaches based on better process management and the use of good software engineering practice.
* The SEI process maturity framework identifies maturity levels that essentially correspond to the use of good software engineering practice.