SA, LI E01963939

Chapter 2 Software Processes Specification-defining what the system should do; Design and implementation –defining the organization of the system and implementing the system; Validation-checking that it does what the customer wants; Evolution-changing the system in response to changing customer needs.

A model is a simplified representation of a software process.

1. waterfall-plan-driven model 2. Incremental dev.-specification, development and validation are interleaved.3. Integration & configuration what type of components are frequently re-used?

2.2 Process Activities: Software engineering as an evolutionary process Two types revolutionary models: Prototyping and Spiral models.

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. Benefits of prototyping ″ Improved system usability. ″ A closer match to users’ real needs. ″ Improved design quality. ″ Improved maintainability. ″ Reduced development effort.

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 prioritized 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 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.

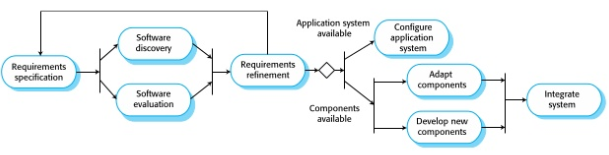
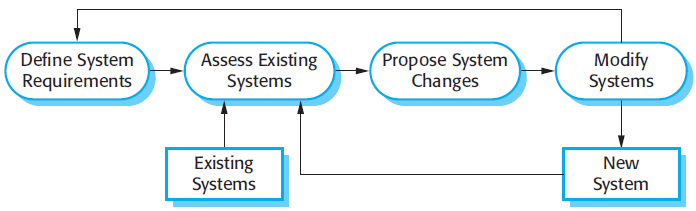
Incremental development benefits ″ The cost of accommodating changing customer requirements is reduced. ♣ The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model. ″ It is easier to get customer feedback on the development work that has been done. ♣ Customers can comment on demonstrations of the software and see how much has been implemented. ″ More rapid delivery and deployment of useful software to the customer is possible. ♣ Customers are able to use and gain value from the software earlier than is possible with a waterfall process. Reuse-oriented software engineering Key process stages ″ Requirements specification ″ Software discovery and evaluation ″ Requirements refinement ″ Application system configuration ″ Component adaptation and integration Advantages and disadvantages ″ Reduced costs and risks as less software is developed from scratch ″ Faster delivery and deployment of system ″ But requirements compromises are inevitable so system may not meet real needs of users ″ Loss of control over evolution of reused system elements

Process activities ″ Real software processes are inter-leaved sequences of technical, collaborative and managerial activities with the overall goal of specifying, designing, implementing and testing a software system. ″ The four basic process activities of specification, development, validation and evolution are organized differently in different development processes. ″ For example, in the waterfall model, they are organized in sequence, whereas in incremental development they are interleaved.

System implementation ″ The software is implemented either by developing a program or programs or by configuring an application system. ″ Design and implementation are interleaved activities for most types of software system. ″ Programming is an individual activity with no standard process. ″ Debugging is the activity of finding program faults and correcting these faults.

Software prototyping ″ A prototype is an initial version of a system used to demonstrate concepts and try out design options.

Benefits of prototyping ″ Improved system usability. ″ A closer match to users’ real needs. ″ Improved design quality. ″ Improved maintainability. ″ Reduced development effort.



Chapter 3 Agile development ″ Program specification, design and implementation are inter-leaved ″ The system is developed as a series of versions or increments with stakeholders involved in version specification and evaluation ″ Frequent delivery of new versions for evaluation ″ Extensive tool support (e.g. automated testing tools) used to support development. ″ Minimal documentation – focus on working code

 A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance. Not necessarily waterfall model – plan-driven, incremental development is possible Iteration occurs within activities. ″ Agile development Specification, design, implementation and testing are inter- leaved and the outputs from the development process are decided through a process of negotiation during the software development process.

Extreme Programming (XP) takes an ‘extreme’ approach to iterative development. New versions may be built several times per day; Increments are delivered to customers every 2 weeks; All tests must be run for every build and the build is only accepted if tests run successfully.

Refactoring ″ Conventional wisdom in software engineering is to design for change. It is worth spending time and effort anticipating changes as this reduces costs later in the life cycle. ″ XP, however, maintains that this is not worthwhile as changes cannot be reliably anticipated. ″ Rather, it proposes constant code improvement (refactoring) to make changes easier when they have to be implemented.

Refactoring ″ Programming team look for possible software improvements and make these improvements even where there is no immediate need for them. ″ This improves the understandability of the software and so reduces the need for documentation. ″ Changes are easier to make because the code is well- structured and clear. ″ However, some changes requires architecture refactoring and this is much more expensive Examples of refactoring ″ Re-organization of a class hierarchy to remove duplicate code. ″ Tidying up and renaming attributes and methods to make them easier to understand. Test-driven development ″ Writing tests before code clarifies the requirements to be implemented. ″ Tests are written as programs rather than data so that they can be executed automatically. The test includes a check that it has executed correctly. Usually relies on a testing framework such as Junit. ″ All previous and new tests are run automatically when new functionality is added, thus checking that the new functionality has not introduced errors.

Pair programming ″ Pair programming involves programmers working in pairs, developing code together. ″ This helps develop common ownership of code and spreads knowledge across the team. ″ It serves as an informal review process as each line of code is looked at by more than 1 person. ″ It encourages refactoring as the whole team can benefit from improving the system code. Pair programming ″ In pair programming, programmers sit together at the same computer to develop the software. ″ Pairs are created dynamically so that all team members work with each other during the development process. ″ The sharing of knowledge that happens during pair programming is very important as it reduces the overall risks to a project when team members leave. ″ Pair programming is not necessarily inefficient and there is some evidence that suggests that a pair working together is more efficient than 2 programmers working separately.  Scrum ″ Scrum is an agile method that focuses on managing iterative development rather than specific agile practices. ″ There are three phases in Scrum. The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture. This is followed by a series of sprint cycles, where each cycle develops an increment of the system. The project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project. Scrum terminology (a) Scrum term Definition Development team A self-organizing group of software developers, which should be no more than 7 people. Agile methods are incremental development methods that focus on rapid software development, frequent releases of the software, reducing process overheads by minimizing documentation and producing high-quality code. ″ Agile development practices include User stories for system specification Frequent releases of the software, Continuous software improvement Test-first development Customer participation in the development team. Scrum is an agile method that provides a project management framework. It is centred round a set of sprints, which are fixed time periods when a system increment is developed. ″ Many practical development methods are a mixture of plan-based and agile development.