Advanced Software Engineering
(CS6401)

Autumn Semester (2024-2025)

Dr. Judhistir Mahapatro
Department of Computer Science and Engineering
National Institute of Technology Rourkela

Topics covered in Previous Lecture:

- Waterfall model of development
- Iterative waterfall
- Incremental
- Evolutionary model
- V-moel
- Prototyping
- Spiral

Rapid application development (RAD)

- It was developed to overcome the rigidity (difficult to accommodate the change of requests from the customer) of the waterfall model (and its variants).

- It includes the features of both prototyping and evolutionary models (where requirements gathering is not fully done).

Rapid application development (RAD)

- It deploys an evolutionary delivery model to obtain and incorporate the customer feedbacks on incrementally delivered versions.

- Prototypes are constructed, and incrementally the features are developed and delivered to the customer. Unlike prototyping model, prototypes are not thrown away but are enhanced and used in the software construction.

RAD Model

- Major goals:
  - To decrease the time taken and the cost incurred to develop software systems.
  - To limit the costs of accommodating change requests.
  - To reduce the communication gap between the customer and the developers.

RAD Model

- It is established among the practitioners that only through commenting on an installed application that the exact requirements can be brought out.

RAD Model

- In iterative waterfall model
  - Often clients do not know what they exactly wanted until they saw a working system.
  - Customers do not get to see the software until the development is complete and deployed.
  - Delivered software do not meet the customer expectations and changes are incorporated through subsequent maintenance efforts.
  - Cost of accommodating the changes extremely high and took long time to have a good solution

RAD Model

- RAD model tries to overcome this problem by inviting and incorporating customer feedback on successively developed and refined prototypes.

Working of RAD

- Development takes place in a series of short cycles or iterations.
- Development team focuses on the present iteration only and plans are made for only one increment at a time.
- The time planned for each iteration is called a time box.
- Each iteration is planned to enhance the implemented functionality of the application by only a small amount.

Working of RAD

- During each time box, a quick-and-dirty prototype-style software for some functionality is developed.
- Customer evaluates the prototype and gives feedback on specific improvements.
- Prototype is refined based on the customer feedback. However, it is not meant to be released to the customer for regular use.

Working of RAD

- Development team includes a customer representative to clarify the requirements.
- Customer suggest changes to a specific feature already delivered and they have used it.
- Delivery of an incremental feature saves cost
  - this is carried out before large investments have been made in development and testing of large number of features.

Working of RAD

- Decrease in development time and cost, and an increased flexibility to incorporate changes
  - Minimal use of planning
  - Heavy reuse of any existing code through rapid prototyping.

Working of RAD

- RAD model emphasizes code reuse
- Developers earliest to embrace object-oriented languages and practices.
- RAD advocates use of specialized tools to facilitate fast creation of working prototypes.
- Specialized tools support the following features:
  - Visual style of development.
  - Use of reusable components.

Applicability of RAD Model

- Characteristics of an application that indicate its suitability to RAD style of development
  - **Customized software:**
    - It is developed for one or two customers
    - Substantial reuse is usually made of code from pre-existing software.
    - For example, Automatic data processing activities such as registration and grading at one or more educational institutes. When any other institute requests for an automation package in which a few aspects to be tailored, Projects involving such tailoring can be carried out speedily and cost-effectively.

Cont…

- **Non-critical software:**
  - a quick-and-dirty software developed is usually far from being optimal in performance and reliability.
- **Highly constrained project schedule:**
  - It aims to reduce development time at the expense of good documentation, performance and reliability.
  - It is preferred for projects with very aggressive time schedules
- **Large Software:**
  - Software having many features can have incremental development and delivery be meaningfully carried out.

Characteristics that render RAD unsuitable

- **Generic products (wide distribution)**
  - Software products are generic in nature and usually have wide distribution.
  - Optimal performance and reliability are imperative in a competitive market.
- **Requirement of optimal performance and/or reliability**
  - Categories of products require performance or reliability
    - Example, Operating System (high reliability required)

Characteristics that render RAD unsuitable

- **Lack of similar products:**
  - Developing company has not developed similar software, it would hardly be able to reuse much of the existing artifacts.
  - Non-availability of sufficient plug-in components, it becomes difficult to develop rapid prototypes through reuse
- **Monolithic entity:**
  - Especially small-sized software, it is hard to divide the required features into parts that can be incrementally developed and delivered.

RAD versus Prototyping model

- The code developed during prototype construction is usually thrown away. In contrast, developed prototype evolves into the deliverable software.

- Though RAD is expected to lead to faster software development compared to the traditional models (such as prototyping model), but the quality and reliability would be inferior.

RAD versus iterative waterfall model

- All the functionalities of a software are developed together whereas functionalities are developed incrementally through heavy code and design reuse
- In RAD, prototype is refined based on customer feedback. Whereas, iterative waterfall model does not accommodate change requests.
- Important advantages of Iterative Waterfall Model:
  - Use of iterative model leads to production of good quality documentation that helps during maintenance.
  - Developed software usually has better quality and reliability

RAD versus evolutionary model

- In RAD, each increment results essentially a quick and dirty prototype
- In evolutionary model, each increment is systematically developed using the iterative waterfall model.
- In RAD, software is developed in much shorter increments compared to the evolutionary model.

Agile development model

- Over last two decades, rapid shift from development of software products to development of customized software
  - Increased emphasis on scope for reuse.

- It was primarily designed to help a project to adapt to change requests quickly. (overcome the shortcomings of waterfall model)

Agile development model

- Aim of it is to facilitate quick project completion.
- How? Agility is achieved by fitting the process to the project, i.e., removing activities that may not be necessary for a specific project.
- Anything that wastes time and effort is avoided.

Popular agile SDLC models

- Crystal
- Atern
- Feature-driven development
- Scrum
- Extreme programming
- Lean development
- Unified process

Cont...

- Agile model adopts an iterative approach.
- Each iteration lasts for a couple of weeks only.
- Time to complete an iteration is called time box.
- End date for an iteration does not change.
  - Team may reduce functionality during a time box if necessary.

Essential Idea behind Agile Models

- It is suited for development of small projects.
- In case of large projects, collaborating teams working at different locations are needed to have arranged as many as meetings through video conferencing.
- The agile model emphasizes incremental release of working software as the primary measure of progress.

Essential Idea behind Agile Models

Important Principles:

- Working software over comprehensive documentation.
- Frequent delivery of incremental versions of the software to the customer in intervals of few weeks.
- Requirement change requests from the customer are encouraged and are efficiently incorporated.
- Enhanced communication among the team members can be realized through face-to-face communication rather than through exchange of formal documents.

Essential Idea behind Agile Models

Important Principles:

- Continuous interaction with the customer is considered much more important than effective contract negotiation.

- Agile development projects usually deploy pair programming where two programmers work together at one work station.
  - One types in code while the other reviews the code.
  - The two programmers switch their roles every hour or so.

Advantages and Disadvantages

- Agility by relying on the tacit knowledge of the team members about the development project and informal communication to clarify issues rather spending time in preparing formal documents and reviewing them.

- Lack of adequate documentation may lead to several problems:
  - Scope for confusion and important decision taken during different phases can be misinterpreted at later points
  - It becomes difficult for design decisions to be reviewed by external experts.
  - When the project completes and the developers disperse, maintenance can become a problem.

Agile versus RAD model

- Agile model does not recommend developing prototypes, but emphasizes systematic development of each incremental feature. In contrast, quick-and-dirty prototypes are then refined into production quality code.

- Agile projects logically break down the solution into features that are incrementally developed and delivered. In contrast, developing all the features by improving successively over time.

Agile versus RAD model (Cont..)

- Agile team only demonstrates completed work to the customer. In contrast, development teams demonstrate to customers screen mock ups and prototypes.

Extreme programming model

- It recommends taking these best practices that have worked well in the past in program development projects to extreme levels.
  - If something is known to be beneficial, why not put it to constant use?
  - Based on this principle, it puts forward several key practices that need to be practiced to the extreme.

Extreme programming model

- XP is based on frequent releases (called iterations), during which the developers implement “user stories”. It is similar to use cases, but are more informal and are simpler.
- Example, a user story about a library software can be:
  1. a library member can issue a book.
  2. A library member can query about the availability of a book.

Extreme programming model

- Project team proposes a common vision of how the system would work.
  - Essentially, it's the Architectural design

- Team may decide to construct a Spike for some features.
  - Spike: Very simple program constructed to explore the suitability of a solution being proposed.
  - Its similar to Prototype.

Good practices recognized

- Code review:
  - helps detect and correct errors most efficiently.
  - Pair programming as the way to achieve continuous review. While one writes other reviews the code
- Testing:
  - Suggests Test-driven development, where test cases are written even before any code is written.
- Incremental development:
  - It helps to get customer feedback

Good practices recognized

- Simplicity:
  - Simplest code ignores the aspects such as efficiency, reliability, maintainability, etc.
  - Once the basic functionality works then these aspects can be introduced through refactoring.
- Design:
  - Everybody should design daily.
  - Quality design can be achieved through refactoring (improve the nonfunctional attributes (such as readability, and maintainability) of the software without affecting the external behavior).

Good practices recognized

- Integration Testing:
  - Helps identify bugs at the interfaces
  - Developers should achieve continuous integration by building and performing integration testing several times a day.

XP prescribes several basic activities

Coding:

- Utmost care and attention needed for coding activity
- Without coding it is not possible to have a working system
- Coding in XP has a slightly different meaning.
- Example: coding activity includes drawing diagrams (modeling) that will be transformed to code, scripting a web-based system and choosing among several alternative solutions.

XP prescribes several basic activities

Testing:

- High importance for testing and considers it as a primary means for developing a fault-free software

Listening:

- Developers need to listen to the customers carefully
- Customers usually have the domain knowledge but programmers may not have in-depth knowledge of the domain

XP prescribes several basic activities

Designing:

- Without good design implementation becomes too complex, and dependencies within the system become too numerous, and it becomes very difficult to comprehend the solution.
- Maintenance difficult and expensive
- Emphasizes use of a suitable design technique

Feedback:

- Understanding the exact requirement
- Feedback is critical to learning and making changes
- Frequent contact with the customer makes the development effective

XP prescribes several basic activities

Simplicity:

- Build something simple that will work today rather something taking a lot of time never getting used
- Attention on making specific functions that are immediately needed

Applicability of XP model

- Projects involving new technology or research projects
  - requirement change rapidly and unforeseen technical problems need to be resolved.
- Small projects
  - face to face meeting is easier to achieve.

Characteristics not suited to use of XP model

- Stable requirements
  - Almost no change to the gathered requirements
- Mission critical or safety critical systems
  - It can not ensure the required reliability

Reference

- R. Mall, *Fundamentals of Software Engineering*, Prentice Hall of India , 2014