Software Process and Models

Instructor: Dr. Jerry Gao
Software Process Models

- Software Process

- Software Process Models
  - Waterfall Model
  - Prototyping Model
  - RAD Model

- Evolutionary Software Process Models
  - Incremental Model
  - Spiral Model
  - Component Assembly Model
  - Concurrent Development Model

- Process Maturity
- Process and Product
Software Process

- A common software process framework:
  - can be used to all software projects.
  - a number of task sets, including tasks, milestones, deliverables, and SQA points.
- Umbrella activities:
  - project tracking and control
  - technical reviews (formal or informal)
  - software quality assurance (SQA)
  - configuration management
  - documentation
  - measurement
  - management of reuse and risk
Software Process Models

Concepts:
- a process model or a software engineering paradigm:
  --> refers to a strategy to encompasses the process, methods, and tools
together to effectively manage and deliver a software product.

Select a process model based on:
- the nature of the project and application
- the methods and tools to be used
- the controls and deliverables

There are four fundamental process activities:
- Software specification.
- Software development
- Software validation
- Software maintenance (or evolution)
All software development can be viewed as a software problem solving loop. It consists of four stages:
- problem definition - identify problems and define them
- technical development - find the solutions to solve them
- solution integration - apply solutions and deliver the system
- status QUO - current state of affairs.
The ‘Waterfall’ Model

The waterfall model - the linear sequential model, also called the “classic life cycle” ---> a systematic, sequential approach to software development that begins at the system level and progress through analysis, design, coding, testing, and maintenance.
The Waterfall Model

The waterfall model is the oldest and most widely used paradigm for software engineering.

Advantages: Simple, step-by-step, focused, and easy to follow.

Problems:

- Inflexible because real projects rarely follow the sequential flow that the model proposes.

- It is often difficult for the customers to state all requirements explicitly.

- The customer must have patience to wait to validate the software product in the late phases. (until programs are implemented)

- Customers are involved the project in the beginning.

- Developers are often delayed unnecessarily between phases.
The Prototyping Model

- step 1: Requirements gathering
- step 2: A “quick design” -> focuses on visible functions and behaviors of the product
- step 3: Prototype construction
- step 4: Customer evaluation of the prototype
  loop back step 1.

Listen to customer

Build/revise Mock-up

Customer test-drives mock-up
The Prototyping Model

Advantages:
- Easy and quick to identify customer requirements
- Customers can validate the prototype at the earlier stage and provide their inputs and feedback
- Good to deal with the following cases:
  - Customer can not provide the detailed requirements
  - Very complicated system-user interactions
  - Use new technologies, hardware and algorithms
  - Develop new domain application systems

Problems:
- The prototype can serve as “the first system”. Brooks recommends we throw away.
- Developers usually attempt to develop the product based on the prototype.
- Developers often makes implementation compromises in order to get a prototyping working quickly.
- Customers may be unaware that the prototype is not a product, which is held with.
The RAD Model

Rapid Application Development (RAD) is a linear sequential software development process model that emphasize an extremely short development cycle.
- A “high-speed” adaptation of linear sequential model
- Component-based construction
- Effective when requirements are well understood and project scope is constrained.

Advantages:
- Short development time
- Cost reduction due to software reuse and component-based construction

Problems:
- For large, but scalable projects, RAD requires sufficient resources.
- RAD requires developers and customers who are committed to the schedule.
- Constructed software are project-specific, and may not be well modularized.
- Its quality depends on the quality of existing components.
- Not appropriate projects with high technical risk and new technologies.
The RAD Model

- **Business Modeling**: Modeling information flow among functions

- **Data Modeling**: Modeling information flow in terms of business data and objects

- **Process Modeling**: Defining functional processes by mapping the two previous results

- **Application Generation**: Use fourth generation techniques or automatic tools to construct software

- **Testing & Turnover**: Testing new components perform integration tests

Team # i

60-90 days
The RAD Model

Rapid Application Development (RAD) is a linear sequential software development process model that emphasize an extremely short development cycle.
- A “high-speed” adaptation of linear sequential model
- Component-based construction
- Effective when requirements are well understood and project scope is constrained.

Advantages:
- Short development time
- Cost reduction due to software reuse and component-based construction

Problems:
- For large, but scalable projects, RAD requires sufficient resources.
- RAD requires developers and customers who are committed to the schedule.
- Constructed software are project-specific, and may not be well modularized.
- Its quality depends on the quality of existing components.
- Not appropriate projects with high technical risk and new technologies.
Evolutionary Software Process Models

Classic process models are not designed to deliver a production system due to their assumptions on:
- A complete system will be delivered after the linear sequence is completed.
- Customer knows what they want at the early stage.

The reality in a software production process -->
- A lot of requirements changes during the production course
- A lot of iterative activities and work because of the evolutionary nature of software production

To cope with the product evolution, several evolution process models are proposed:
- the incremental model
- the spiral model
- the component assembly model
- the concurrent development model
The Incremental Model

The Incremental Model combines elements of the linear sequential model with the iterative philosophy of prototyping. The first increment is a core product.

The incremental process model:
- Iterative in nature, like prototyping.
- Focus on the delivery of a operational product with each increment.
- Particular useful when staffing is unavailable for a complete implementation by the business deadline.
The Spiral Model

The spiral model (by Boehm[BOE88])
- is an evolutionary software process model
- couples the iterative nature of prototyping
- include the systematic aspects of the linear sequential model.
- provides the potential for rapid development of incremental versions of the software.

The spiral model is divided into a number of framework activities (regions):
- customer communication
- planning (resources, timelines, etc.)
- risk analysis
- engineering
- construction & release
- customer evaluation

Each region is populated by a series of work tasks.
The Spiral Model

The spiral model (by Boehm[BOE88])
- is an evolutionary software process model
- couples the iterative nature of prototyping
- include the systematic aspects of the linear sequential model.
- provides the potential for rapid development of incremental versions of the software.

The spiral model is divided into a number of framework activities (regions):
- customer communication
- planning (resources, timelines, etc.)
- risk analysis
- engineering
- construction & release
- customer evaluation

Each region is populated by a series of work tasks.
The spiral model
-- a realistic approach to the development of large scale systems and software.
- may be difficult to convince customers that the evolutionary approach is controllable.
- a relatively new model, and has not used as widely as the linear sequential or prototyping paradigms.
The Component Assembly Model

Planning

Risk Analysis

Customer Communication

Customer Evaluation

Engineering Construction & Release

Identify candidate component

Construct nth iteration of system

Look up components in library

Put new components in library

Extract components if available

Build components if unavailable
The Component Assembly Model

Object Technologies -- the technical framework for a component-based process model for software engineering.

The component assembly model
- incorporates many of the characteristics of the spiral model
- is an evolutionary in nature
- demand an iterative approach to the creation of software
- leads to software reuse, and reusability

Advantages:
- demand software reuse
  --> cost reduction
  --> reduction in development cycle time

What is the limitation of the component assembly model?
What are the current problems in this model?
The Concurrent Development Model

- Under development
- Under review
- Under Revision
- Baselined
- Done
- Awaiting changes

none
The Concurrent Development Model

The concurrent development model - called concurrent engineering. --> provides an accurate state of the current state of a project.

- Focus on concurrent engineering activities in a software engineering process such as prototyping, analysis modeling, requirements specification and design.

- Represented schematically as a series of major technical activities, tasks, and their associated states.

- Defined as a series of events that trigger transitions from state to state for each of the software engineering activities.

Two ways to achieve the concurrency:

- system and component activities occur simultaneously and can be modeling using the state-oriented approach

- a typical client/server application is implemented with many components, each can be designed and realized concurrently.

Application: all types of software development
Software Process Maturity

Five process maturity levels:

Level 1: Initial - Software process --> ad hoc, and occasionally even chaotic.

Level 2: Repeatable - Basic project management processes are established to track cost, schedule, and functionality.

Level 3: Defined - In software process, both management and engineering activities is documented, standardized, and integrated into an organization-wide software process.

Level 4: Managed - Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using detailed measures.

Level 5: Optimizing - Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies.