Parul® University

Advanced Software Engineering

Dr. Kapil Aggarwal
Associate Professor
Computer Science & Engineering

UNIT - 8

Advanced Software Engineering

Parul® University

DIGITAL LEARNING CONTENT

Advanced Software Engineering Topics

- Software Reuse
- Component Based Software Engineering
- Distributed Software Engineering
- Service-Oriented Software Engineering
- Real-Time Software Engineering
- Systems Engineering
- Systems of System

Software Reuse

Reuse-based software engineering is an approach to development that tries to maximize the reuse of existing software.

- Availability of reusable software at low cost
- Demands for lower software production and maintenance costs
- Faster delivery of systems
- Recognized software quality

- **Software Reuse Levels**

  - Application system reuse
  - Component reuse
  - Object and function reuse

Benefits of Software Reuse

- **Increased dependability**
  - Tried and tested
  - More dependable than new software
  - Identified and Fixed design and implementation faults
- **Reduced process risk**
  - Known cost of existing software
  - Easy to make decision for Project Rebuilt
  - Reduces the margin of error in project cost estimation
  - Highly Helpful for large software components-subsystems

Benefits of Software Reuse

- **Effective use of specialists**
  - Reduce Rework
  - Better Utilization specialist's knowledge
- **Standards compliance**
  - Use of standard user interfaces
  - Improves dependability
  - Reduces mistakes from user as familiar interface

Image source : Google

Parul® University

Benefits of Software Reuse

- **Accelerated development**
  - Launch a system to market at earliest
  - Speed up system production
  - Reduction in development and validation time

Threats or Challenges with Software Reuse

- Lack of tool support
- Not-invented-here syndrome
- Creating, maintaining, and using a component library
- May Increased maintenance costs
- Finding, understanding, and adapting reusable components

Parul® University

Key factors during planning Software Reuse

- Development schedule
- Expected software lifetime
- Development team's background, skills, and experience
- Criticality of the software
- Software's non-functional requirements
- The application domain
- The execution platform

Component Based Software Engineering (CBSE)

- Components and Component Models
- Component Based Software Engineering Processes
- Component Formation

Component and Component Models

- **Components - Highly Useful for Achieving Reusability**
  - Module which can be independently functions deployed or composed without modifications from the system.
- **Characteristics of Components**
  - Standardized, Independent, Compos-able, Deployable, Documented
- **Component Models:**
  - Based on definition of standards for component implementation, documentation, and deployment.
  - Implement through interfaces, usage and Deployment through Platform or Support Services

Component Based Software Engineering Processes

- **Types of Component Based Software Engineering Processes**
  - Development for reuse
  - Development with reuse
- **Reuse supports processes related to**
  - Component acquisition
  - Component management
  - Component certification

Component Formation

- **Component Composition / Formation:**
  - The process of integrating components
- **Types of Component Composition:**
  - Sequential composition
  - Hierarchical composition
  - Additive composition
- **Factors in Failure of Component Composition**
  - Parameter Incompatibility & Operation Incompatibility
  - Operation incompleteness

Distributed Software Engineering

- **Distributed Systems – Presently Known as Cloud**
  - A collection of independent computers
  - Appears to the user as a single coherent system
- **Software Engineering Practices are different for Distributed Systems**
- **Main Focuses handling following during development:**
  - Distributed systems issues
  - Client–server computing
  - Architectural patterns for distributed systems
  - Software as a service

Distributed systems issues

- **Important design issues that have to be considered**
  - Transparency
  - Openness
  - Scalability
  - Security
  - Quality of service
  - Failure management
- **Dimensions of Scalability – Size, Distribution, Manageability**
- **Types of attacks –Interception, Interruption, Modification, Fabrication**
- **The quality of service reflects the system's ability**
- **Recovery Plans - Models of interaction and Middleware**

Client–server computing

- **Distributed systems that are referred as client–server systems**
- **To create and process that information**
  - Depend on Various layers for computations
- **Have layered architectural model for client–server application**
  - Presentation layer
  - Data management layer
  - Application processing layer
  - Database layer

Architectural patterns for distributed systems

- **Applicable architectural styles:**
  - Master-slave architecture
  - Two-tier client–server architecture
  - Multitier client–server architecture
  - Distributed component architecture
  - Peer-to-peer architecture

Software as a service

- **Useful to reduce client side dependencies and need as requirement of client-server application**
- **Examples are Oauth Service, Google Docs, Sheets, One Notes etc.**
- **Also popular as SaaS - Software as a Service**
  - Software is deployed on a server
  - The software is owned and managed
  - Users may pay for the software requiring to the amount of use
- **Important factors into consideration during development**
  - Configurability
  - Multi-tenancy
  - Scalability

Parul® University

DIGITAL LEARNING CONTENT

Service-Oriented Software Engineering

- **Focused on the development of software systems by composition of**
  - Reusable services
  - Separation of concerns
- **Extends characteristics of component-based software engineering**
- **Attention to :**
  - Service-oriented interaction pattern
  - Service-oriented analysis and design

Service-Oriented Software Design Process

- **Primary Concerns are Focuses on:**
  - Service candidate identification
  - Service interface design
  - Service implementation and deployment
  - Legacy system services
- **Service construction by composition:**
  - Workflow design and implementation
  - Service testing

Real-Time Software Engineering

- **Time Critical Response and Result Required Systems (Time Constrained)**
  - Soft Real-Time Systems – Some Delays Permitted
  - Hard Real-Time Systems – No Delays Permitted
- **Used to Monitor and Control Environments/Systems/Hardware**
- **Example – IoT / Embedded Based System**
- **Stimulus/Response Systems**
  - Periodic stimuli
  - Aperiodic stimuli

Real-Time Software Systems Design Process

- **Real-Time Systems design key factors**
  - Real-time programming
  - Real Time Process management
  - Real-time Operating Systems
- **Attention to:**
  - Process Priority, Switching, Scheduling and Interrupt handling

Systems Engineering

- **Interdisciplinary field of engineering and engineering management**
- **System Engineering Emphases on How to following on complex systems over their life cycles :**
  - Design
  - Integrate
  - Manage complex systems over their life cycles
- **Systems engineering handles:**
  - Work-processes, optimization methods, and risk management tools in such projects

Systems in System Engineering

- **System Engineering Tools:**
  - Strategies
  - Procedures
  - Techniques
- **System Engineering Models**
  - An abstraction of reality designed to answer specific questions about the real world, through an imitation, analogue, or representation of a real world process or structure represented in conceptual, mathematical, or physical tool to assist a decision maker.

Parul® University

DIGITAL LEARNING CONTENT

System Engineering Process

- **Task definition**
  - Informative definition
- **Conceptual stage**
  - Cardinal definition
- **Design stage**
  - Formative definition
- **Implementation stage**
  - Manufacturing definition

Systems of System

- **Collection of capable of independent functioning systems**
  - Example: Enterprise Software
- **Goal:**
  - Collected Systems interoperate together to achieve additional desired capabilities
- **Have communication structure among system**
- **Types of Systems of Systems**

Types of Systems of Systems

- **Virtual**
  - Lack a central management authority and a centrally agreed-on purpose
- **Collaborative**
  - Interact more or less voluntarily to fulfill agreed-on central purposes
- **Acknowledged**
  - Recognized objectives and a designated resources
- **Directed**
  - Built and centrally managed during long-term operation

References

1 Pressman, Roger S. "Software engineering: A professional approach." (2016).

2 Sommerville, Ian. "Software engineering 9th Edition." ISBN-10 137035152 (2011).

DIGITAL LEARNING CONTENT

Parul® University