Major Exam II Reschedule

5:30 – 7:30 pm in Sat Dec 9nd

Recall The Team Skills

1. Analyzing the Problem (with 5 steps)
2. Understanding User and Stakeholder Needs
3. Defining the System
4. Managing Scope
5. **Refining the System Definition**
    1. Software Requirements: a more rigorous look
    2. Refining the Use cases
    3. Developing the Supplementary Specification
    4. On Ambiguity and Specificity
    5. Technical Methods for Specifying Requirements
6. Building the Right System

Chapter 22
Developing the Supplementary Specification

- Role of the Supplementary Specification
- Nonfunctional Requirements
- Design Constraints
- Linking the Supplementary Spec. to the Use Cases

The Role of the Supplementary Specification

- There are many requirements can't be implemented as in the use-case model.
- Example: "The application must run on Windows XP"
- So there is a need for supplementary specification.
- Supplementary: because we assume the use-case format will contain most of the functional requirements for the system, and we'll supplement the use-case model with these additional requirements.

Exploring Nonfunctional Requirements

- Nonfunctional requirements can be organized into four categories:
  - Usability
  - Reliability
  - Performance
  - Supportability

Usability (*ease of use*)

1. Specify the required training time for a user to become
   - minimally productive: able to accomplish simple tasks) and
   - operationally productive: able to accomplish normal day-to-day tasks).
2. Specify measurable task times for typical tasks or transactions that the end user will be carrying out.
3. Compare the usability of the new system with other state-of-the-art systems that the user community knows and likes.

Usability (*ease of use*)

4. Specify the existence and required features of online help systems, wizards, tool tips, user manuals, and other forms of documentation and assistance.

5. Follow conventions and standards that have been developed for the human-to-machine interface.

Reliability

1. Availability. The system must be available for operational use during a specified percentage of the time.
2. Mean time between failures (MTBF). This is usually specified in hours, but it also could be specified in days, months, or years.
3. Mean time to repair (MTTR). How long is the system allowed to be out of operation after it has failed?

Reliability

4. Accuracy. What precision is required in systems that produce numerical outputs?
5. Maximum bugs, or defect rate. This is usually expressed in terms of bugs/KLOC (thousands of lines of code) or bugs per function-point.
6. Bugs per type. This is usually categorized in terms of minor, significant, and critical bugs.

Performance

1. Response time for a transaction: average, maximum
2. Throughput: transactions per second
3. Capacity: the number of customers or transactions the system can accommodate
4. Degradation modes: the acceptable mode of operation when the system has been degraded

Supportability

- Supportability is the ability of the software to be easily modified to accommodate enhancements and repairs.
- Example : "Modifications to the system for a new set of withholding tax rates shall be accomplished by the team within 1 day of notification by the tax regulatory authority."

Understanding Design Constraints

- Design constraints are restrictions on the design of a system, or the process by which a system is developed, that do not affect the external behaviour of the system but that must be fulfilled to meet technical, business, or contractual obligations.
- Sources of Design Constraints
  1. Restriction of design options
  2. Conditions imposed on the development process
  3. Regulations and imposed standards.

Restriction of Design Options

- Most requirements allow for more than one design option.
- Whenever possible, we want to leave that choice to the designers rather than specifying it in the requirements.
- Whenever we do not allow a choice to be made ("Use Oracle DBMS"), the design has been constrained, and a degree of flexibility and development freedom has been lost.

Conditions Imposed on the Development Process

- These are due to the following:
- Compatibility with existing systems: "The application must run on both our new and old platforms."
- Application standards: "Use the class library from Developer's Library 99-724 on the corporate IT server."
- Corporate best practices and standards: "Compatibility with the legacy database must be maintained." "Use our C++ coding standards."

Regulations and Imposed Standards

- Typical regulatory design constraints might include regulations and standards from the following:
  - Food and Drug Administration (FDA)
  - Federal Communications Commission (FCC)
  - Department of Defence (DOD)
  - International Organization for Standardization (ISO)
  - Underwriters Laboratory (UL)
  - International standards such as the German Industrial Standard (DIN)

Handling Design Constraints

- Distinguish them from the other requirements.
- Include all design constraints in a special section of your requirements, or use a special attribute so they can be readily aggregated.
- Identify the source of each design constraint.
- Document the rationale for each design constraint.

Are Design Constraints True Requirements?

- Yes, they do meet the definition of a requirement as something necessary to satisfy a contract, standard, specification, or other formally imposed documentation.

- Treat a design constraint just like any other requirement and make certain that the system is designed and developed in compliance with that design constraint.

Identifying Other Requirements

- Physical artifacts (CDs and so on) that are deliverables of the system
- Target system configuration and preparation requirements
- Support or training requirements
- Internationalization and localization requirements

Linking the Supplementary Specification to the Use Cases

- How do non-functional requirements apply to the use cases?
- Do specific use cases have associated non-functional requirements, and, if so, how could we indicate that?

Linking the Supplementary Specification to the Use Cases

- One way to do so is to define certain classes of non-functional requirements. For example, we might define "Quality of Service" classes for response time as follows:
  - Class 1: 0 to 250 milliseconds
  - Class 2: 251 to 499 milliseconds
  - Class 3: 0.5 to 2 seconds
  - Class 4: 2.1 to 12 seconds
  - Class 5: 12.1 seconds to 60 minutes

Linking the Supplementary Specification to the Use Cases

- Then we could associate these classes with special requirements recorded in the use case itself. For example, Use Case A might record
  - Response time: Class 2 for main flow of events, Class 4 for all exceptions
- special requirements is an additional section that can be included in the documentation of a use case.

- You can do the same for other classes of non-functional requirements (such as reliability, safety, and so on) and map these requirements to the specific use cases.

Template for the Supplementary Specification

1. **Introduction**
   - **1.1. Purpose**
     State the purpose of the document (to collect all functional requirements not expressed in the use-case model, as well as nonfunctional requirements and design constraints).
   - **1.2. Scope**
   - **1.3. Definitions, Acronyms, and Abbreviations**
   - **1.4. References**
2. **Functional Requirements**
   Describe the functional requirements of the system for those requirements that are expressed in the natural language style or are otherwise not included in the use-case model.
3. **Usability**
   State the requirements that affect usability.
4. **Reliability**
   State the requirements for reliability.
5. **Performance**
   State the performance characteristics of the system, expressed quantitatively where possible and related to use cases where applicable.
6. **Supportability**
   State the requirements that enhance system supportability or maintainability.

Template for the Supplementary Specification *(Cont'd)*

7. **Design Constraints**
   State the design or development constraints imposed on the system or development process.

8. **Documentation Requirements**
   State the requirements for user and/or administrator documentation.

9. **Purchased Components**
   List the purchased components used with the system, licensing or usage restrictions, and compatibility/ interoperability requirements.

10. **Interfaces**
    Define the interfaces that must be supported by the application.

    10.1. User Interfaces

    10.2. Hardware Interfaces

    10.3. Software Interfaces

    10.4. Communications Interfaces

Template for the Supplementary Specification *(Cont'd)*

11. **Licensing and Security Requirements**
    Describe the licensing and usage enforcement requirements or other restrictions for usage, security, and accessibility.

12. **Legal, Copyright, and Other Notices**
    State any required legal disclaimers, warranties, copyright notices, patent notices, trademarks, or logo compliance issues.

13. **Applicable Standards**
    Reference any applicable standards and the specific sections of any such standards that apply.

14. **Internationalization and Localization**
    State any requirements for support and application of different user languages and dialects.

15. **Physical Deliverables**
    Define any specific deliverable artifacts required by the user or customer.

16. **Installation and Deployment**
    Describe any specific configuration or target system preparation required to support installation and deployment of the system.