CSE414: WEB ENGINEERING

Daffodil international university

Learning Outcomes

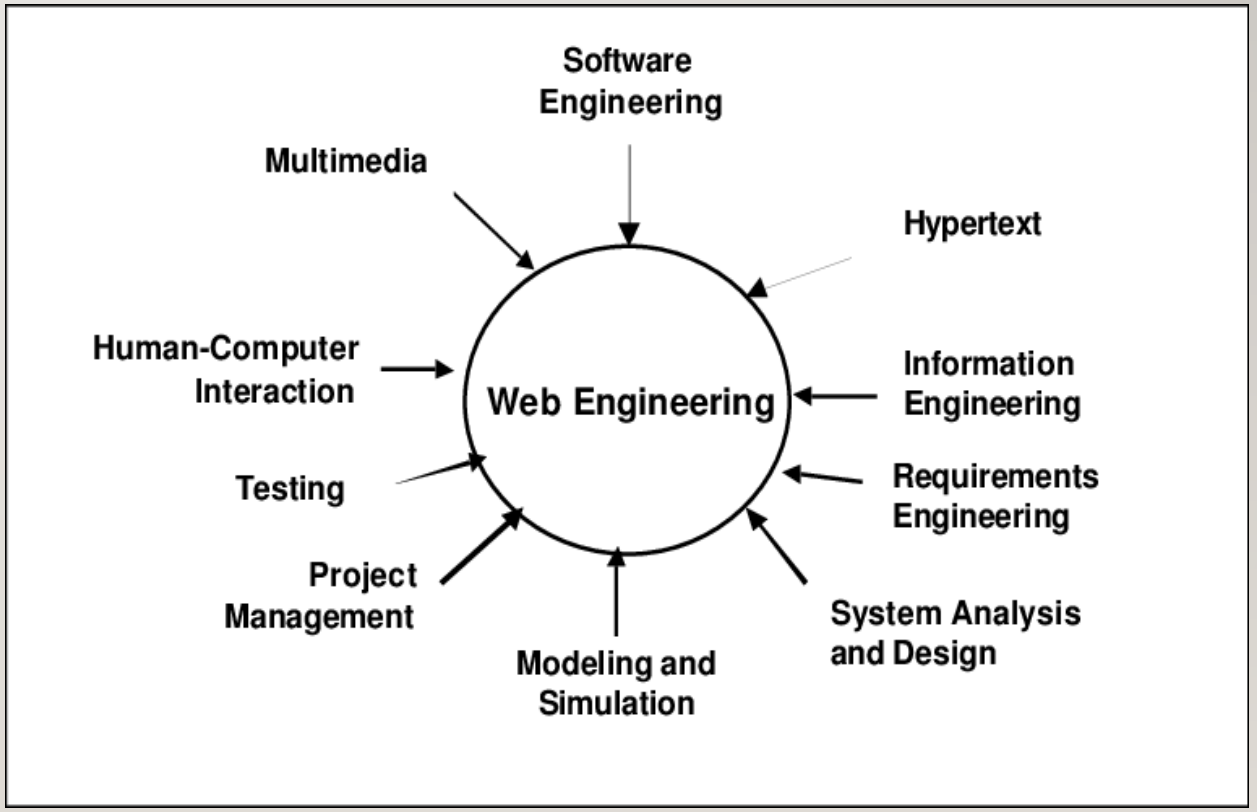
- Analyze System Requirements For,
  - Existing Systems
- Design Requirement Specification For,
  - New Systems
- Choose Right Testing Methodology

Contents

- Requirement Engineering
- Testing Methodology

Software Engineering VS Web Engineering

- You already know about software engineering from CSE333!
- So, what's new about web engineering?
- **Good news**:
  - This lecture reviews a lot from SWE!
- On a different note, you will see the power of these theories in upcoming days! We use them all!



Is there any difference or relation?

- "*In theory, there is no difference between theory and practice. But, in practice, there is.*"

- **Software engineering** is an engineering discipline that is concerned with all aspects of software production.

- **Web Engineering** is the application of systematic, disciplined and quantifiable approaches to development, operation, and maintenance of Web-based applications.

What differences has the web made to Software Engineering?

- Availability of software services
- Developing highly distributed service-based systems
- Led to important advances in-
  - Programming languages
  - Software reuse

Requirement Engineering

- The requirements for a system are the descriptions of what the system should do— the services that it provides and the constraints on its operation.
- Requirements should describe **what** the system should do, but **not how** it should do it.
- The process of finding out, analyzing, documenting and checking these services and constraints is called requirements engineering (RE).

Requirement Engineering

- Two kinds of requirements based on the intended purpose and target audience:
  - User Requirements – what services the system is expected to provide to system users and the constraints under which it must operate.
  - System Requirements – more detailed descriptions of the software system's functions, services, and operational constraints.

User requirements & System requirements

- User requirements
  - High-level abstract requirements
  - Written as statements, in a natural language plus diagrams,
  - Services the system is expected to provide to system users
  - Constraints under which it must operate.
- System requirements
  - Detailed description of system
  - Functions, services, and operational constraints.
  - The system requirements document (sometimes called a functional specification) should define exactly what is to be implemented.

Requirement Engineering

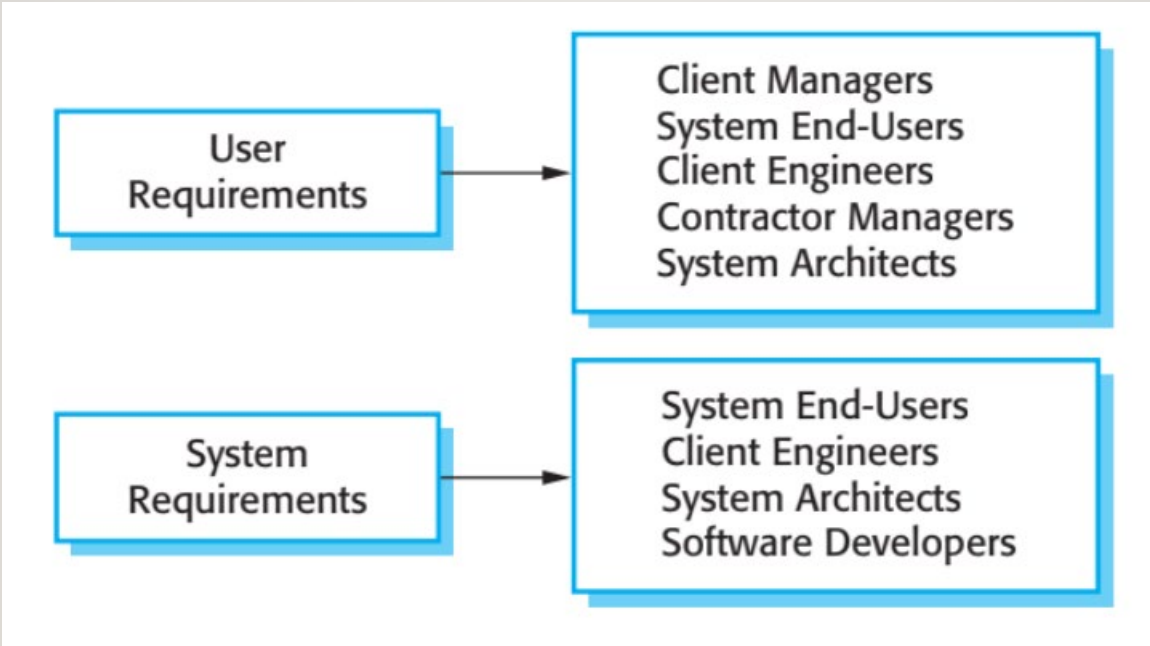


Fig: Readers of different types of requirements specification

Requirement Engineering

- Software system requirements are often classified as
  - Functional requirements – statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations.
  - Non-functional requirements – These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process, and constraints imposed by standards.

Functional Requirements

- Describe **functionality** or system services
  - ✓ Depends on the type of software, expected users and the type of system where the software is used
    - ➢ May be high-level statements of what the system should do
    - ➢ Should describe the system services in detail
- Problems arise when-
  - Requirements are not precise and interpreted in different ways by developers and users.
- Requirements should be both
  - **Complete** : include descriptions of all facilities required
  - **Consistent** : should be no contradictions in descriptions
- In **practice** , it is impossible to produce a complete and consistent requirements document

Non-functional Requirements

- System properties and constraints
  - e.g. reliability, response time and storage requirements
- Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular IDE, programming language or development method.
- Non-functional requirements may affect the overall architecture
  - A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
- It may also generate requirements that restrict existing requirements.

Non-Functional Requirements

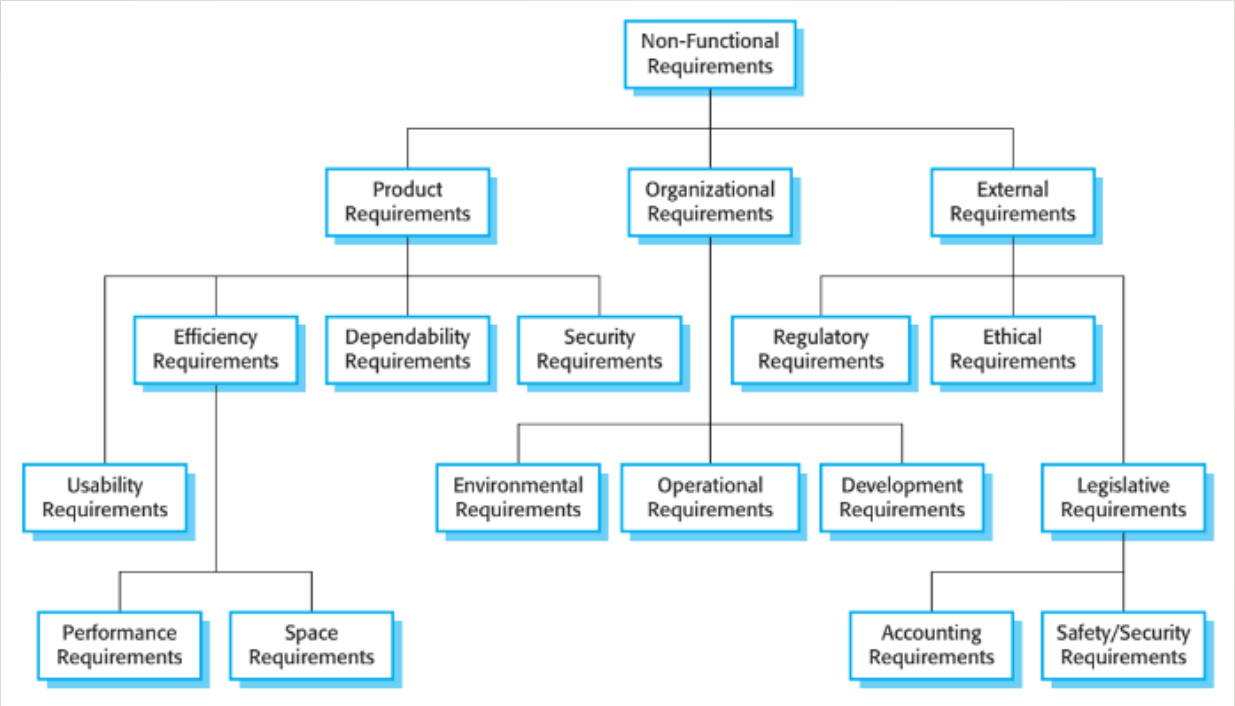


Fig: Types of non-functional requirement

Non-Functional Requirements

Three classes of non-functional requirements :

- Product requirements
  - Requirements which specify that the delivered product must behave in a particular way
    - e.g. execution speed, reliability, memory, security requirements, usability requirements etc.
- Organizational requirements
  - Requirements which are a consequence of organizational policies and procedures
    - e.g. process standards used, implementation requirements, etc.
- External requirements
  - Requirements which arise from factors which are external to the system and its development process
    - e.g. interoperability requirements, legislative requirements, etc.

Non-Functional Requirements

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Metrics for specifying non-functional requirements

Domain Requirements

- The system's operational domain imposes requirements on the system.
- Domain requirements may be new functional requirements, constraints on existing requirements or define specific computations.
- If domain requirements are not satisfied, the system may be unworkable.
- Two main **problems** :
  - Understandability
    - ✓ Requirements are expressed in the language of the application domain, which is not always understood by software engineers developing the system.
  - Implicitness
    - ✓ Domain specialists understand the area so well that they do not think of making the domain requirements explicit.

Requirements Engineering Process

- Processes vary widely depending on the application domain, the people involved and the organization developing the requirements.
- In practice, requirements engineering is an **iterative process** in which the following generic activities are interleaved:
  - Requirements **elicitation** ;
  - Requirements **analysis**;
  - Requirements **validation** ;
  - Requirements **management** .

Elicitation and Analysis

- Range of system **stakeholders**
  - Requirements discovery by interacting with stakeholders
  - Requirements are grouped and organized into coherent clusters
  - Prioritizing requirements and resolving requirements conflicts
  - Requirements are documented and input into the next round of the spiral
- Open **interviews** with stakeholders are a part of the RE process.
- **User stories** and **scenarios** can be used and easy for stakeholders to understand
  - i.e., Use-case diagram

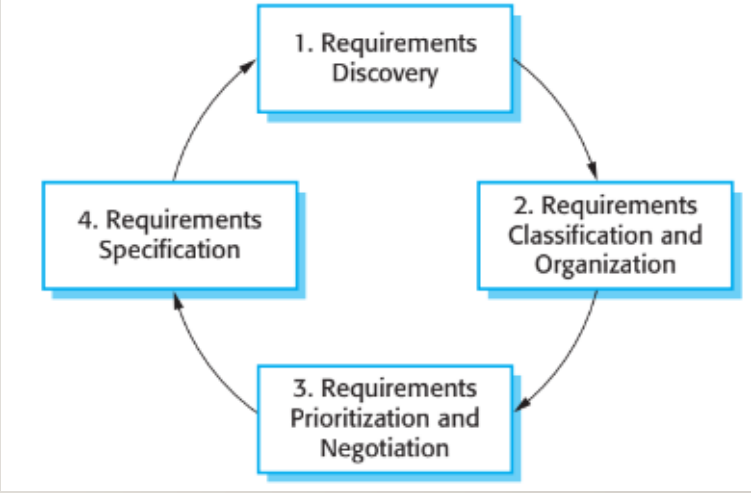


Fig: Requirements elicitation and analysis process

Elicitation and Analysis

- **Problems** :
  - **Stakeholders don't know what they really want.**
  - Stakeholders express requirements in their own terms.
  - Different stakeholders may have conflicting requirements.
  - Organizational and political factors.
  - Requirements change during the analysis process.

Requirements Specification and Validation

- **Requirements specification**
  - Writing down user and system requirements in a requirements document
  - Written in **natural language** supplemented by **appropriate diagrams** and **tables**
  - **Structured natural language** is used to maintain a standard way.
- **Requirements Validation**
  - Demonstrates that the requirements define the system that the customer really wants
  - Regular reviews should be held while the requirements definition is being formulated
  - Using an executable model of the system to check requirements
  - Developing tests for requirements to check testability

Requirements Change

- Challenge
  - Requirements are ever changing
- New requirements emerge
  - As the system being developed
  - After it has gone into use
- Reasons why requirements change after the system's deployment:
  - Business and technical environment always changes
  - Customers and Users are two different group people
  - Large systems usually have a diverse user community
    - different requirements and priorities that may be conflicting or contradictory.

Testing

- **Testing** is intended
  - to show that a **program does what it is intended to do**
  - to **discover program defects** before it is put into use.
- **Testing can reveal the presence of errors, but NOT their absence**
- Testing is part of a more general verification and validation process
- Error
  - the actual result deviates from the expected
    - Our expected results should (theoretically) come from our requirements definition
    - Most often, the goals/concerns/expectations of stakeholders serve as the testing basis.

Testing

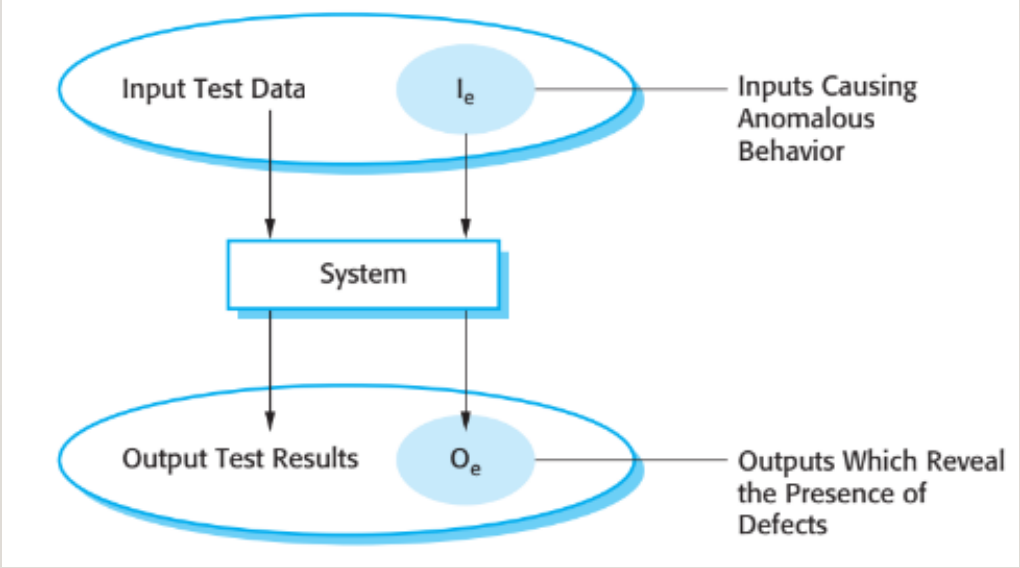


Fig: An input-output model of program testing

Testing

- **Test Case**
  - a set of inputs, execution conditions, and expected results for testing an object
- Complete test coverage is impossible, so testing focuses on mitigating the largest risks.
  - Where's the greatest potential for loss?
  - What are the sources of this risk?
- **Start testing as early as possible** – even with restricted resources and time.

Goals of Software Testing

- To **demonstrate** to the developer and the customer that the **software meets its requirements** .
  - Leads to **validation testing** :
    - you expect the system to perform correctly using a given set of test cases that reflect the system's expected use.
  - A successful test shows that the system operates as intended.
- To **discover** situations in which the behavior of the software is **incorrect, undesirable or does not conform to its specification** .
  - Leads to **defect testing** : the test cases are designed to expose defects; the test cases can be deliberately obscure and need not reflect how the system is normally used.
  - A successful test is a test that makes the system perform incorrectly and so exposes a defect in the system.

Verification and Validation

- **Verification : Are we building the product right?**
  - The software should conform to its specification.
- **Validation : Are we building the right product?**
  - The software should do what the user really requires.
- The aim of verification is to check that the software meets its stated functional and non-functional requirements.
- The aim of validation is to ensure that the software meets the customer's expectations.

Aims of Verification and Validation

- Establish confidence that the system is **good enough for its intended use**, which depends on:
  - **Software purpose** : the level of confidence depends on how critical the software is to an organization.
  - **User expectations** : users may have low expectations of certain kinds of software.
  - **Marketing environment** : getting a product to market early may be more important than finding defects in the program.

Three Stages of Testing

- **Development testing** : the system is tested during development to discover bugs and defects.
- **Release testing**: a separate testing team test a complete version of the system before it is released to users.
- **User testing** : users or potential users of a system test the system in their own environment.

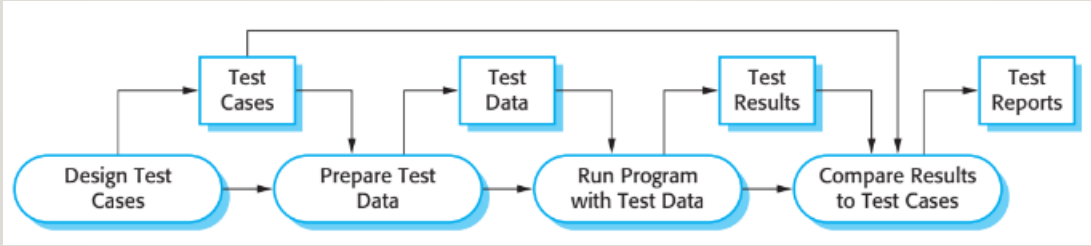


Fig: A model of testing process

Development Testing

Development testing includes all testing activities that are carried out by the team developing the system:

- **Unit testing** :
  - individual program units or object classes are tested; should focus on testing the functionality of objects or methods.
- **Component testing** :
  - several individual units are integrated to create composite components; should focus on testing component interfaces.
- **System testing** :
  - some or all of the components in a system are integrated and the system is tested as a whole; should focus on testing component interactions.

Release Testing

- **Release testing**
  - testing a particular release
- **Intended** for **use outside of the development team**
- **Convince** the **customer** of the system that it is **good enough** for use.
- **Requirements -based testing**
  - examining each requirement and developing a test or tests for it
  - demonstrate that the system has properly implemented its requirements
- **Scenario testing**
  - devise typical scenarios of use and use these to develop test cases for the system
  - Scenarios should be realistic and real system users should be able to relate to them.
  - If scenarios were developed as part of the requirements engineering process, it can be reused as testing scenarios.

User Testing

User or customer testing is a stage in the testing process in which **users or customers provide input and advice on system testing** .

- User testing is essential, even when comprehensive system and release testing have been carried out. Types of user testing include:

❖ **Alpha testing** :

- ➢ Users of the software work with the development team to test the software at the developer's site.

❖ **Beta testing** :

- ➢ A release of the software is made available to users to allow them to experiment and to raise problems that they discover with the system developers.

❖ **Acceptance testing** :

- ➢ Customers test a system to decide whether or not it is ready to be accepted from the system developers and deployed in the customer environment.

Test Methods And Techniques

- **Link Testing :** Finding broken links, orphan pages etc.
- **Browser Testing :** variety of browsers exists. Thus need to test.
- **Load Testing :** Does the system meet required response times and throughput?
- **Stress Testing :** How does the system behave under abnormal/extreme conditions?
- **Continuous Testing :** Typically, running the operation a few times doesn't produce an error, hence the need for continuous testing.
- **Security Testing :**
  - Is our SSL certificate working?
  - What happens if I try to access a protected page/site in a non-secure way (i.e., http://)?

Exercise And Readings

- **EXERCISE**
  - *Develop SRS for you course project (soft copy submission)*
  - Write a short notes on
    - Web Project Management
    - Testing tools.
- **READINGS**
  - Requirement Engineering
    - https://cs.ccsu.edu/~stan/classes/CS410/Notes16/04-Requirements.html
  - Software Testing
    - https://cs.ccsu.edu/~stan/classes/CS410/Notes16/08-SoftwareTesting.html
  - Project Management
    - https://cs.ccsu.edu/~stan/classes/CS410/Notes16/22-ProjectManagement.html
  - Testing tools
    - https://medium.com/@briananderson2209/best-automation-testing-tools-for-2018-top-10-reviews-8a4a19f664d2

References

- Requirements Engineering
  - Reference: Sommerville, Software Engineering, 10 ed., Chapter 4
- Testing
  - Sommerville, Software Engineering, 10 ed., Chapter 8
- Project Management
  - Sommerville, Software Engineering, 9 ed., Chapter 22
- https://www.researchgate.net/figure/Web-Engineering-A-multidisciplinary-field_fig1_220795871
- https://www.quora.com/What-is-web-engineering