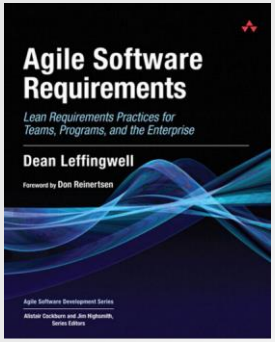


Agile Software Requirements

Software Requirements Engineering – 40688

Computer Engineering department

Sharif university of technology

Fall 402

Chapter 17:

Nonfunctional Requirements

FURPS

- *Functionality:* What the system does for the user
- *Usability:* How easy it is for a user to get the system to do it.
- *Reliability:* How reliably the system does it
- *Performance:* How often, or how many of it, it can do.
- *Supportability:* How easy it is for us to maintain and extend the system that does it.
- Placeholder for organizing the NFRs,

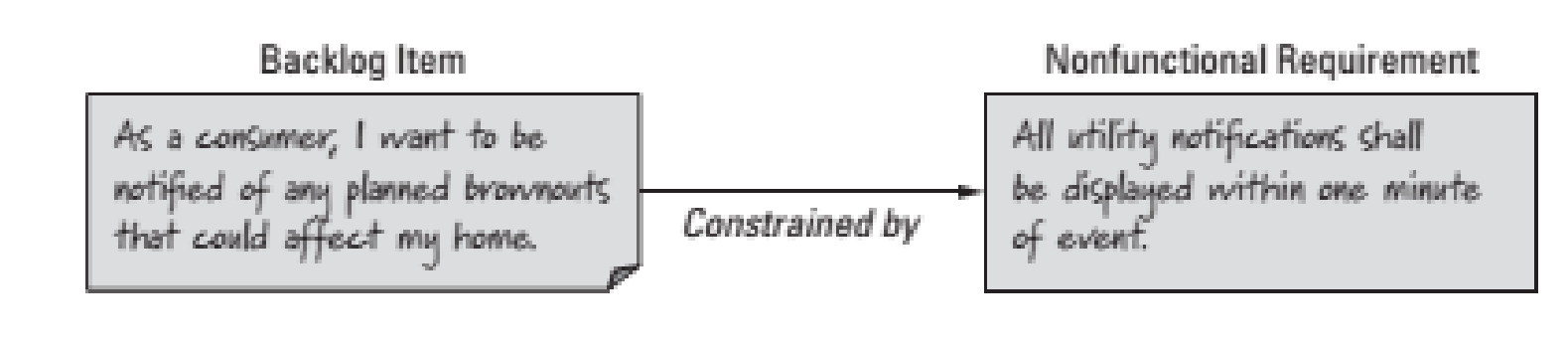
Requirement Types and Descriptions

Requirement Type	Description	Examples
Functional requirements	Express how the system interacts with its users—its inputs, its outputs, and the functions and features it provides.	Display a pop-up on the TV when the utility sends a brownout warning.
Nonfunctional requirements	Criteria used to judge the operation or qualities of a system.	The system must be available to its users at least 99.99% of the time. The systems should support 100 concurrent users with no degradation in performance.
Design constraints	Restrictions on the design of a system, or the process by which a system is developed, but that must be fulfilled to meet technical, business, or contractual obligations.	Use Python for all client applications. Don't use any open source software that doesn't conform to the GNU General Public License.

Modeling Nonfunctional Requirements

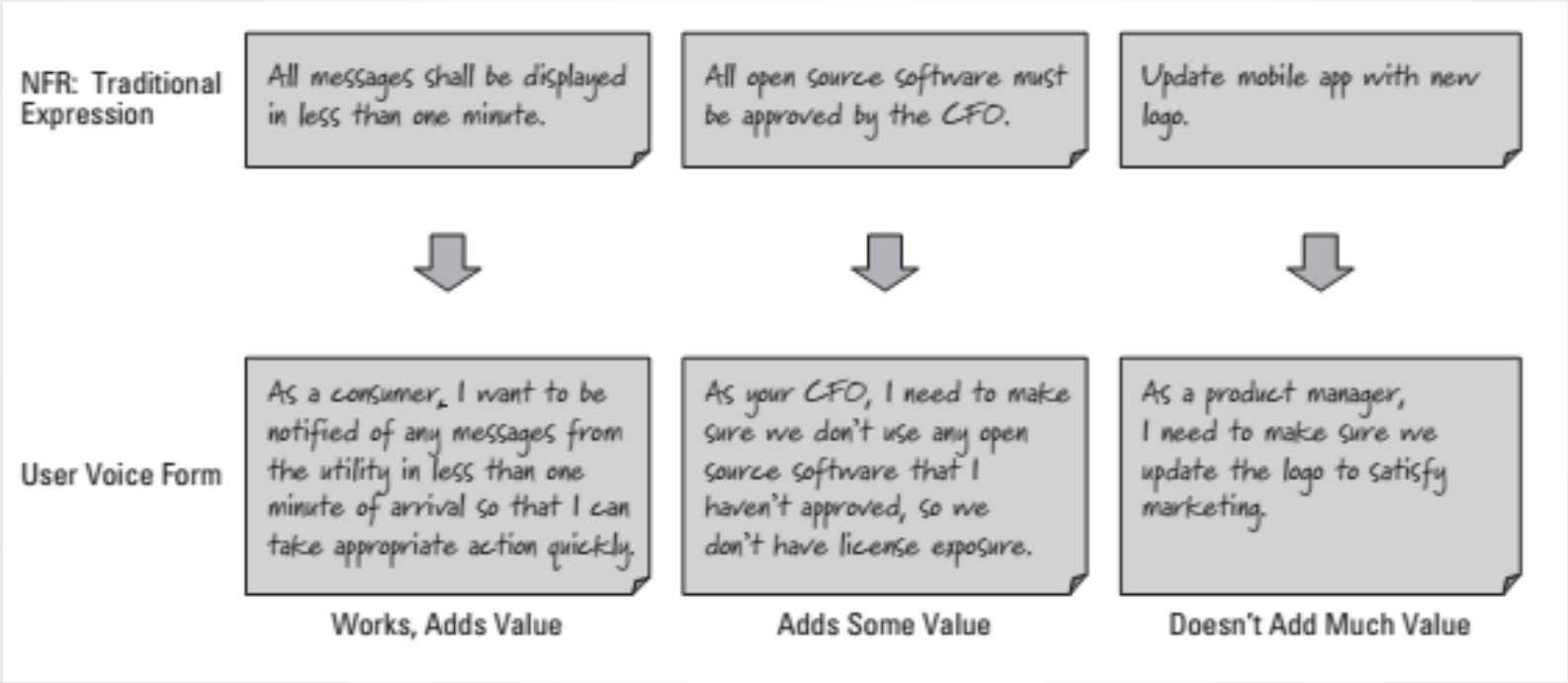
Association between backlog items and nonfunctional requirements

Modeling Nonfunctional Requirements



User story constrained by a nonfunctional requirement

User Story Voice



Usability

- Specify the training time objective for a user.
- Specify measurable task times for typical tasks or transactions that the end user will be carrying out.
- Compare the user's experiences with other comparable systems that the user community knows and likes.
- Specify any required user assistance features such as online help, wizards, tool tips, context-sensitive help, user manuals, and other forms of documentation and assistance.
- 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.
- **Mean time between failures (MTBF):** This is usually specified in hours, but it also could be specified in days, months, or years.
- **Mean time to repair (MTTR):** How long is the system allowed to be out of operation after it has failed?

Reliability (2)

- **Accuracy**: What accuracy is required in systems that produce numerical outputs?
- **Defects**: Defects may be categorized in terms of minor, significant, and critical.
- **Security**: Application security is of paramount importance, and it is a critical business priority to design security into the system.

Performance (1)

- **Response time**: Specify for transactions of a given type, average and worst case.
- **Throughput**: Specify in transactions per second, latency, overhead, data transmission rates, and so on.
- **Capacity**: Specify the number of customers, transactions, data, and so on, the system can accommodate.
- **Scalability**: Specify the ability of the system to be extended to accommodate more interactions and/or users.

Performance (2)

- Degradation modes: Define an acceptable behavior for when the system has been degraded.
- Resource utilization: If the new system has to share hardware resources with other systems or applications, it may also be necessary to stipulate the degree to which the implementation will make "civilized" use of such resources as the CPU, memory, channels, disk storage, and bandwidth.

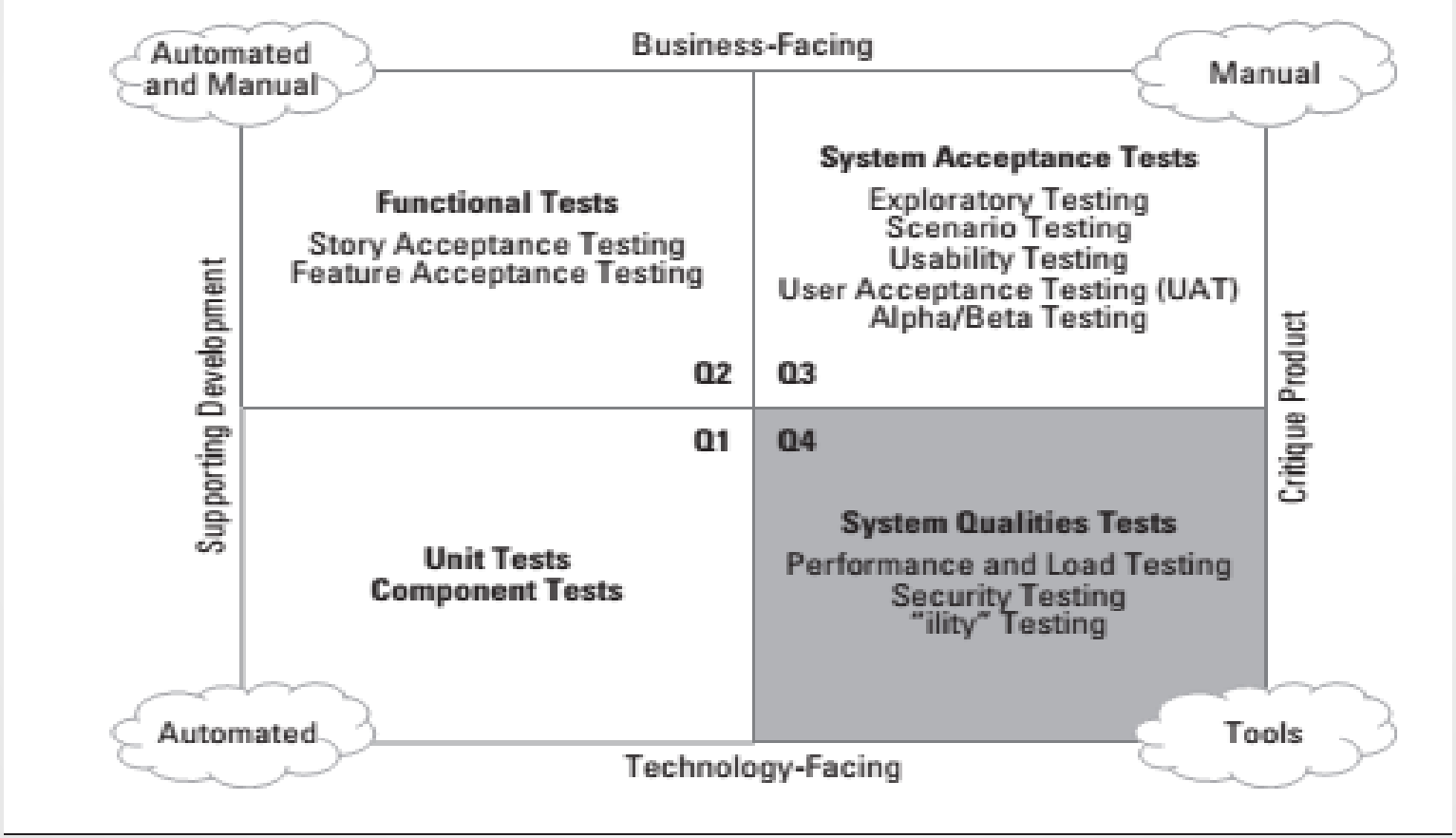
Supportability (Maintainability)

- Supportability (maintainability) includes the ability of the software to be easily modified to accommodate enhancements and repairs.
- For some application domains, the likely nature and even timing of future enhancements can be anticipated (protocol changes, annual tax rule changes, standards compliance response timelines, availability of new data sources, and so on).
- There may be a mandate for a team to be able to respond to these anticipated changes.

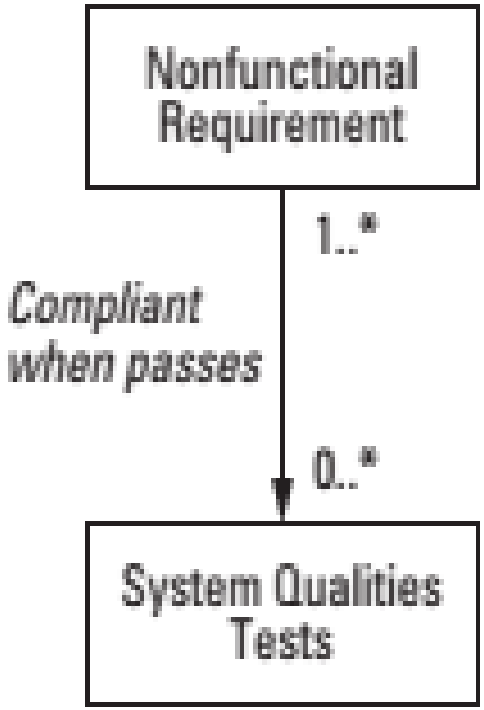
Testing nonfunctional requirements

- Whether these nonfunctional requirements are testable.
- The answer is assuredly yes, because most all of these constraints (performance, for example) can and should be objectively tested.

Testing Matrix



Testing


Nonfunctional Requirement
1..*
Compliant when passes
0..*
System Qualities Tests

Usability (1)

- Usability testing is a black-box testing technique.
- Tests how easy it is for users to achieve their objectives with the system.
- Typically, usability testing involves gathering a small number of users (three to five) together and then having them execute scripts that use the system in predetermined ways.
- Focus on measuring four aspects

Usability (2)

- **Productivity**: How long does it take a user to perform a particular task?
- **Accuracy**: How many errors or missteps does the user experience along the way?
- **Recall**: How easily can a user recall how to use the system if they have been away from the system for a while?
- **Emotional response**: How does the user react to the experience of using the system—drudgery, acceptable, interesting, or fun?

Reliability

- Reliability is somewhat easier to test because the objectives may be clearer,
- Especially if the team has stated any service-level requirements such as availability, mean time, between failures
- There are many language-specific profiling tools that assist developers in performing low-level
- Tests such as testing for memory leaks, potential race, and other code conditions that have shown to be typical root causes of reliability problems.
- Load and Stress

Security

- Can be approached from two perspectives, white box testing and black-box testing.
- In white-box testing, the testing regime examines the actual code to look for potential coding practices and paths through the code that can allow security breaches.
- Black-box testing mimics the way in which real-life hackers try to defeat a system.
- Using scripts and tools, the test regime can inject various faulty inputs into the system and try to “break the system.”

Performance

- Performance testing is usually done with the assistance of specialized tools.
- At the scope of system-level testing, user and other system load simulators, measuring, and monitoring tools are available.
- These are used to simulate a heavy load on a server, network, system component, or other object to test its resilience and to analyze overall performance under different load types.

1. Introduction

1.1. Purpose

To record all nonfunctional requirements for the system.

1.2. Scope

1.3. Definitions, Acronyms, and Abbreviations

1.4. References

2. Usability

State any requirements that affect usability, and link them to specific domains of functionality where applicable.

3. Reliability

State any requirements for reliability, quantitatively wherever possible.

4. Performance

State any performance requirements of the system, expressed quantitatively where possible, and link to specific features or user stories where applicable.

6. Supportability

State any requirements for system supportability or maintainability.

7. Design Constraints

State any 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 any purchased components used with the system, licensing or usage restrictions, and compatibility/interoperability requirements.