Requirements

15-413: Introduction to Software Engineering

Jonathan Aldrich

Some slides taken from Ian Sommerville, Software Engineering 7

Functional Requirements

- What the machine should do
  - Input
  - Output
  - Interface
  - Response to events
- Criteria
  - Completeness
    - All requirements are documented
  - Consistency
    - No conflicts between requirements
  - Precision
    - No ambiguity in requirements

Quality Requirements

- Specify not the functionality of the system, but the quality with which it delivers that functionality
- Quantify wherever possible
  - Requirements serve as contracts: should be testable/falsifiable
- Can be more critical than functional requirements
  - Can work around missing functionality
  - Low-quality system may be unusable

Example Quality Requirements

Student suggestions

- Performance
- Time guarantees in real time
- Reliability
- Visual appeal / usability
- Scalability
- Security
- Ability to change
  - Code quality

Other Requirements

- Process requirements
- Legal requirements
  - Conformance to law
- Standard compliance
  - Interoperability
- Implementation requirements
  - Language, platform…
- Delivery requirements
  - Format, delivery date, documentation…
- Are these legitimate requirements?
  - Some of these seem like solutions, not problems

Making requirements verifiable

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- Informal goal
  - A general intention of the user such as ease of use.
- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested.
- Informal goals may still be helpful to developers as they convey the intentions of the system users.

Examples

- **Informal goal**
  - The system should be easy to use by experienced controllers and should be organised in such a way that user errors are minimised.
- **Verifiable non-functional requirement**
  - Experienced controllers shall be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users shall not exceed two per day.

Requirements measures

(student suggestions)

Property	**Measure**
Efficiency	Size of the program Throughput Memory footprint User response time
Security	Guard against attacks Pass third-party audit Guard against user error
Usability	Training time Successful user study evaluation (rating >5) User performance on some metric
Reliability	Uptime guarantee based on experimental testing
Evolvability	Experienced programmer outside project carry out some task in X time Conformance to coding/documentation guidelines
Portability	

Stakeholders

- Any person or group who will be affected by the system, directly or indirectly
- Examples
  - End users
  - System administrators
  - Engineers maintaining the system
  - Business managers
- May disagree!
  - Requirements process should trigger negotiation to resolve conflicts

Elicitation Challenges

- Stakeholders don’t know what they want
  - Hard to articulate needs
  - Unaware of cost
- Stakeholder domain knowlege
  - May use jargon
  - May leave out “obvious” domain requirements
    - Not obvious to a software engineer!
- Conflicts among stakeholders
- Politics
  - Managers push for requirements that increase their own influence
- Changes to requirements

Stakeholder Viewpoints

- A framework for organizing stakeholder requirements
  - Benefit: similar viewpoints may have similar requirements
- Interactor viewpoints
  - People who interact with the system
- Indirect viewpoints
  - People who influence the system’s requirements in some other way
- Domain viewpoints
  - Domain characteristics and constraints

Viewpoints Example

(student suggestions)

- Setting: Bank ATM
- Interactor viewpoints
  - Consumer
  - Bank manager (policy settings)
  - People who fill up the ATM
  - Bank server
- Indirect viewpoints
  - Bank manager (policy settings)
  - Federal reserve
  - Thieves or security managers
- Domain viewpoints
  - Laws governing ATM use
  - Agreements with other banks

Interviews

- Deriving requirements from answers to questions in interviews with stakeholders
- Effective interviewers
  - Are open-minded: willing to change ideas in response to stakeholder comments
  - Begin concretely
    - Specific question
    - Requirements proposal
    - Working through a prototype
  - Explore additional issues that come naturally from these questions
    - But staying focused on the system

Interview Benefits and Drawbacks

- Strengths
  - What stakeholders do
  - How they interact with the system
  - Challenges with current systems
- Weaknesses
  - Capturing domain knowledge
    - Familiarity
    - Technical subtlety
  - Organizational issues
    - People may not want to tell the truth about the politics

Scenarios

Source: Ian Sommerville, Software Engineering 7

- Scenarios are real-life examples of how a system can be used.
- Benefits of concrete scenario
  - Easier for non-technical users to relate to than abstract descriptions
  - Criticism draws out details that might otherwise remain implicit
  - Basis for final requirements
- Drawbacks
  - No list of scenarios can cover the full range of situations encountered in practice

Scenarios

- Scenarios should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;
  - Information about other concurrent activities;
  - A description of the state when the scenario finishes.

Library System Scenario (1)

Initial assumption: The user has logged on to the LIBSYS system and has located the journal containing the copy of the article.

Normal: The user selects the article to be copied. He or she is then prompted by the system to either provide subscriber information for the journal or to indicate how they will pay for the article. Alternative payment methods are by credit card or by quoting an organisational account number.

The user is then asked to fill in a copyright form that maintains details of the transaction and they then submit this to the LIBSYS system.

The copyright form is checked and, if OK, the PDF version of the article is downloaded to the LIBSYS working area on the user's computer and the user is informed that it is available. The user is asked to select a printer and a copy of the article is printed. If the article has been flagged as 'print-only' it is deleted from the user's system once the user has confirmed that printing is complete.

Library System Scenario (2)

What can go wrong: The user may fail to fill in the copyright form correctly. In this case, the form should be re-presented to the user for correction. If the resubmitted form is still incorrect then the user's request for the article is rejected.

The payment may be rejected by the system. The user's request for the article is rejected.

The article download may fail. Retry until successful or the user terminates the session.

It may not be possible to print the article. If the article is not flagged as 'print-only' then it is held in the LIBSYS workspace. Otherwise, the article is deleted and the user's account credited with the cost of the article.

Other activities: Simultaneous downloads of other articles.

System state on completion: User is logged on. The downloaded article has been deleted from LIBSYS workspace if it has been flagged as print-only.