ECE155: Engineering Design with Embedded Systems Winter 2013

Lecture 31 — March 26, 2013

Patrick Lam *version 1*

Meeting requirements is critical to the success of a software product. Take ECE451 to learn more.

Requirements

A *requirements specification* is a document that "completely" describes the behaviour of a project. Requirements for software projects can be called software specifications, software requirements specifications (SRS), or just specifications for short. The SRS contains use cases, functional requirements, and nonfunctional requirements.

To get requirements, you iterate *requirements elicitation*, consulting with stakeholders and users. Requirements elicitation includes fact-finding, validating one's understanding of the project, and communicating open issues about the requirements for clarification.

The vision and scope document defines the features of a project, while the requirements specification document defines the behaviour of those features.

Use Cases

The basic unit of a requirements specification is a *use case*. Following *UML Distilled*, we start by defining the notion of a *scenario*[1], which is a sequence of steps describing an interaction between a user and a system.

Example. Describe a scenario of searching for a job in JobMine.

We define a *use case* to be a set of scenarios tied together by a common user goal. Two scenarios associated with the search-for-job JobMine example might be 1) finding a job to apply for and starting the application process; or 2) not finding any jobs to apply for. Often, a use case includes one or several successful paths and a number of failure scenarios.

One way of describing a use case is (as in ASPM) by providing a set of numbered steps for the "main success scenario", and a number of extensions, which are variations of the main scenario.

[1] *Applied Software Project Management*'s use cases are more like scenarios in *UML Distilled*. Tables 6-3 and 6-4 of the *Applied Software Project Management* textbook contain use case examples.

Buy a Product Example. This example is from *UML Distilled*.

Buy a Product

Main Success Scenario:

1. Customer browses through catalog and selects items to buy.
2. Customer goes to check out.
3. Customer fills in shipping information (address; next-day or 3-day delivery).
4. System presents full pricing information, including shipping.
5. Customer fills in credit card information.
6. System authorizes purchase.
7. System confirms the sale immediately.
8. System sends confirming email to customer.

Extensions:

3a: Customer is regular customer.

- 3a1: System displays current shipping, pricing, and billing information.
- 3a2: Customer may accept or override these defaults, returns to MSS at step 6.

6a: System fails to authorize credit purchase.

6a1: Customer may re-enter credit card information or may cancel.

Comments on use cases. The most important part of a use case is the text, which needs to distill the important parts of the use case. Use cases do not describe the user interface nor the implementation, but describe, in general terms, what is supposed to happen as a result of user requests. The complete set of use cases for a system ought to describe all of its functionality.

Use cases often come from external events.

One could also use UML use case diagrams (discussed in L33) to express use cases and relationships between scenarios.

Functional Requirements

Functional requirements augment use cases to fully describe the project's intended behaviour. Functional requirements should be clear, concise and unambiguous (to the intended reader).

Format. Use anything that helps make the requirement clear, including well-chosen words, itemized lists, tables, equations, figures, state diagrams[2], and references to external documents.

Example. What requirements might apply to the JobMine use case above?

Nonfunctional Requirements

You can't see them. *Nonfunctional requirements* describe non-observable properties that the software must satisfy. Also known as non-behavioural requirements or software quality attributes.

Examples. Nonfunctional requirements may specify (among other things) quality, reliability, or a minimum level of performance.

- (software) should not require more than 40 hours of downtime per year for maintenance.
- (network) should not drop more than 1% of call traffic during handoffs between adjacent cells.
- (network) should support at least 12 simultaneous phone calls in each cell.

Functional requirements, on the other hand, specify properties that the software's input must satisfy, as well as what the output must look like.

Software Requirements Specification (SRS)

We've noted that a SRS should include use cases, functional requirements, and nonfunctional requirements. It should also include an introduction and relevant definitions.

Note that use cases and requirements are not the same thing. Use cases don't exhaustively cover requirements, while requirements don't set out to describe all of a user's interactions.

Requirements vs. Design

- Scope: lists project features desired by an organization for its customers ("what", from the buyer's point of view).
- Requirements: specify the behaviour of a project from the perspective of a potential user ("what", from the user's point of view).
- Design: gives high-level implementation details ("how", from the implementer's point of view).

That is, requirements specify external behaviour, or the "essential qualities" of the system, while design describes the structure of the system as we plan to implement it.

[2]The International Telecommunications Union standard for specifying and describing telecommunications systems—the Specification and Description language—is based on communicating Finite State machines.

Changing the Requirements

Changing the requirements can be a big deal. It usually involves more work, hence more cost; and it can have undesired side effects. *Change control* is a way to restrict changes, ideally to include only beneficial changes, thus avoiding project derailment. In particular, a change control board is responsible for reviewing proposed changes and carrying out a cost/benefit analysis.