Monday June 13, 2016

Capturing Requirements

* An activity in the software engineering process
* Transforms Requirements into a specification

Why is it a big deal?

* Clients rarely know precisely what they want
* But you need a precise description to make good software (i.e., correct, usable, on-time, within budget, etc)

Terms

* **Requirement** – in general, an expression of desired behavior (something the system should do or not do)
* **Requirements engineer (RE)/System Analyst** – developer who collects and analyzes requirements, produces, and validates specifications
* **Specification** – Precise description of what software is supposed to do (Software Requirements Specification or SRS)
* **Entity** – a very general term for something that interacts with the system or is acted upon (user type, interface object, external system)

The Information in a Requirement

* Requirements contain one or more of the following

1. Description of behavior (function)
2. Identification of **entity** (user type, interface object, external system)
3. Limitation (constraint) on system, behavior, and/or entity
   1. (i.e., must be built to run on desktop browser and mobile, database)

Requirements Capture Activity

* Elicitation
  + *In requirements engineering, requirements****elicitation****is the practice of collecting the requirements of a system from users, customers and other stakeholders. The practice is also sometimes referred to as "requirement gathering".*
* Analysis
* Specification
  + Structuring the requirements into a consolidated list so you can have all the data that you want in one concise place.
  + “whatever your standards are for the specifications”
* Validation
* Produces

In a Nutshell

* Identify user Stakeholders
* Gather each user stakeholder view of the system
* Analyze each stakeholder statement for behavior, data objects, entities, constraints
* Put analysis into a specifications document and fix problems by going back to the stakeholders or gathering more requirements
* Check for correctness with user stakeholders/fix problems (validation)
* If okay, takes specifications to Planning (tasks, resources, scheduling)

Elicitation Subjects

* Stakeholder types with a viewpoint of the subject domain (1 SH may be more than 1 subject type)
  + **Client** – stakeholder paying for the software
  + **Users** – those who will actually use the software
  + **Domain** **experts** – known the domain and its standards, but maybe not the software
  + **Market** **researchers** – know the customer needs (end-users)
  + **Lawyers** **and** **auditors** – know legal, government, safety requirements
  + **Other** **Software** **Engineers** – can help determining feasibility, estimates

How to elicit requirements

* Interview individual (or group of) stakeholders
* Review available documentation, user’s manuals, handbooks
* Observe current system
* Apprentice with users
* Brainstorm with users to improve proposed system
* Suggest functionality from software with similar features and requirements patterns/templates (i.e., authentication and SCRUD)

Types of Requirements

* **Functional** – behavior, a transformation of data, processing of input into output (usually associated with action verbs. i.e., add user)
* **Non-functional** – Characteristic that software must possess (i.e., huge data, good response time, secure, high reliability)
* **Design** **constraint** – restricts design of system (i.e., target runtime platform, external entity interface or communication protocol; physical constraints of the system itself)
* **Process** **constraint** – restricts software process. Something that changes the software engineering process (i.e., Spiral model to incorporate risk management Stakeholders, Agile methods for early release of some components)

Agile Requirement Process

* Requirement definition and specification can be combined into single document
* Requirement definition is still useful for understanding and validation with non-technical Stakeholders

Requirement Characteristics

Requirements/Specifications should be checked for the following

* **Correct** – RE and client agree requirements matches our understanding
* **Consistent** – No conflicting requirements
* **Unambiguous** – Only one valid interpretation
* **Complete** – Specifies behavior and output for all possible inputs
* **Feasible** – A solution in the requirements is possible
* **Relevance** – Pertains to customer’s real needs
* **Testable** – There is a way to determine if requirements is satisfied
* **Traceable** – Organized and easy to reference