# Mind the Gap: addressing ambiguity in requirements

***Patrick Fleming***
***LANSA North America***

The lesson of the Tower of Babel, possibly the first post-project review in historical records, is that communication failure within the team will cause project failure. In today's projects, often staffed by cross-functional teams spread across the globe, the communication challenge persists.

Complexity in problem definition, solution and design tool architecture, organizational structures and market forces demand agility and constant risk assessment. According to the Project Management Institute (PMI), of the two in five projects that fail to meet business objectives, half traced the cause of failure to ineffective communications. In software development projects these typically include incomplete or changing requirement specifications, and lack of user input.

Don't assume that all project stakeholders have the same understanding of the term 'requirements'. A broad-ranging definition is: Requirements are a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system. A non-functional requirement should define how well the system must do what it does.

'Requirement' in software development projects is also a term overloaded with many different meanings and could be relating to any of the typical project artefacts in the probably incomplete list that follows:

*Business Requirements, Technical Requirements, Stakeholder Requirements, Design Requirements, User Requirements, Functional Requirements, Customer Requirements, Non-functional Requirements, System Requirements, Scope, Process Requirements, High-level Requirements, Detailed-level Requirements, Regulatory Requirements, Product Requirements, Usability Requirements, Quality Requirements, Project Requirements, Data Requirements, Documentation Requirements, Business Rules, Assumptions, Constraints, and more.*

As the classic tree swing cartoon illustrates (originating from the 1970s), all project stakeholders bring their own biases, assumptions, prejudices and expectations which influence the eventual outcome. The IT team or software vendor might be able to define what they can deliver, but only the customer can explain what they need from the system. Choosing which project documents are created and reviewed is based on categorization and criteria, such as:

- By target audience (e.g. stakeholder, user, regulatory and customer requirements)
- By level of detail (e.g. scope level, high level, detailed level and project requirements)
- By business domain (e.g. business rules and business requirements)
- By system/product (e.g. process requirements and product requirements)
- By project (e.g. assumptions, constraints, documentation requirements)
- Technical (e.g. data, design, functional, non-functional, usability, infrastructure requirements)

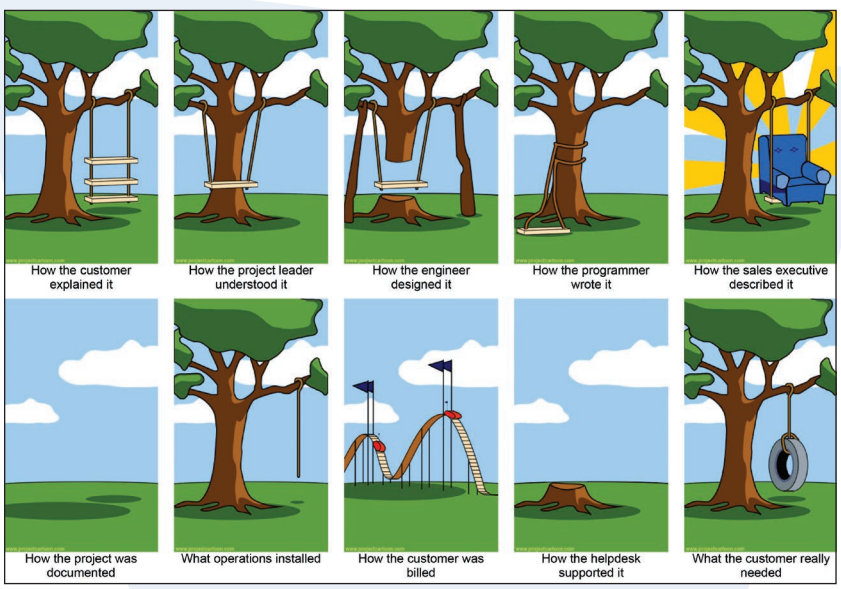
To ensure that the required project deliverables are produced, careful planning and monitoring is needed to manage both the process and the quality of the requirement specifications, in all documentation and review steps.

The planning and monitoring tasks might include:

1. Describe the attributes of a good requirement definition approach, adapted to the business, organization and project.
2. Understand the different kinds of requirements and classify customer input into the appropriate categories.
3. Take an iterative and incremental approach to requirements development.
4. Use standard templates for your vision and scope definition, such as Use Case and System Requirements Specification (SRS) documents.
5. Define the mechanism and process of formal and informal reviews, helping to ensure that stakeholders effectively review, prioritize and explicitly accept the requirements baseline.
6. Instill team and customer discipline to handle requirement changes consistently and effectively.

While Industry groups such as IIBA's BABOK 2.0 and Borland's RDM provide prescriptive standards and processes, we must cater for human frailties in expression and comprehension. Authors often lack formal training in specifying requirements and may use unconstrained natural language which:

- Introduces ambiguity, vagueness and subjectivity
- Is not always clear, concise and coherent
- Is often not testable
- Sometimes misses triggers, implied requirements and exceptions ➔




*Tree Swing Project Management cartoon http://www.projectcartoon.com/cartoon/357611*

Non-standardized business and technical language in English doesn't help. For example, the word 'set' has 194 distinct uses: 58 as a noun, 126 as a verb and 10 as an adjective. The acronym ATM has 129 possible values. Then there are 'doublespeak' examples (from Orwell's book 1984) which deliberately disguise the literal meaning, such as "negative patient care outcome" (death), "career alternative enhancement programs" (layoffs) and "fiscal underachievers" (the poor).

Two important goals of writing requirements are that anyone who reads the requirement should reach the same interpretation as another reader, and each reader's interpretation matches what the author intended to communicate. Language that is explicit, concise and jargon free can be understood by those who don't have the training or the desire to interpret jargon and acronym filled documents.

Take a moment to review the requirements statement column in the table below, and attempt to identify the ambiguity defect. Is there a better way of writing the requirements? Can each requirement be tested as written?

Functional requirements like those in the table can be better expressed from the perspective of something that the system should do, or something the user can do. In some functional requirements, the generic term 'the user' is referred to. The user's role or class or Actor in the application step (e.g. the production manager, the system operator) should be explicitly referenced instead. Also, the term *shall* is preferable to *should* (which may indicate desired), versus *may*, which could mean optional. Semantic nuances between verbs – such as shall, must, may, might, will, would, should, could, needs to, has to, should provide – can make it hard to interpret the requirements consistently.

A single requirement can be matched against objective criteria, measuring how well it is complete, correct, feasible, necessary, prioritized, unambiguous and verifiable. The requirements baseline set collectively should be verified so that the requirements are complete, consistent, modifiable and traceable.

The Easy Approach to Requirements Syntax (EARS) presents a general template for writing concise, testable requirements syntax. It comprises the format: [optional trigger] [optional precondition] Actor Action [Object]. An EARS example requirement specification is: "When an Order is shipped **[trigger]** and Order Terms are not 'Prepaid' **[precondition]**, the system **[Actor]** shall create **[Action]** an Invoice **[Object]**."

The timing, level and precision of the requirements, as specified for developers and testers, can be impacted by whether:

- Work is being done for an internal or external client.
- Developers have considerable domain experience.
- System testing will be based on requirements.
- Accurate estimates are required.
- Precedents are available, such as replacing an existing application.
- Customers are extensively involved.
- Project team members are geographically dispersed.

To discover and resolve ambiguity earlier in the development cycle, have team members who represent diverse perspectives who can formally inspect the requirement documents. Suitable reviewers include:

- the analyst who wrote the requirements
- the customer or representative who supplied them (particularly for use case reviews)
- the developer who must implement them
- the tester who must verify them

Another powerful review technique is to begin writing test cases early on in requirements development. Writing conceptual test cases against both the use cases and functional requirements, reinforces your understanding of how the software should behave under certain conditions. This practice helps reveal ambiguities and missing information, and also leads to a requirements document that can generate comprehensive test cases.

Also consider developing prototypes to help visualize a more tangible result than a text based SRS. Create a preliminary or possible implementation of a poorly understood portion of the requirements, to help identify and clarify gaps in your knowledge. Analysis models such as data flow diagrams, entity-relationship diagrams, class and collaboration diagrams, state-transition diagrams, and dialog maps provide alternative and complementary views of requirements that also reveal knowledge gaps.

The do-nothing alternative could lead to requirement ambiguity, which forces designers and developers to guess. If that happens, who do you want constructing your tree swing? ■

| Functional Requirement Statement | Ambiguity Defect |
|---|---|
| The software shall support a water level sensor. | What does 'support' mean? |
| The thesaurus software shall display about five alternatives to the requested word. | Boundary values – how many is 'about five' 3, 4,6,10 more?<br>Under what conditions are they displayed? |
| Accepted transactions are posted to the database later. Delinquent accounts are reviewed periodically. | When is later?<br>When is periodically? |
| The software shall blink the LED on the adapter using a 50% on 50% off duty cycle. | Does the software blink the LED at all times?<br>Is there a trigger that initiates the blinking. |
| If a boot disk is detected in the system, the software shall boot from it. | What if a boot disk is not present – is that a discrepancy scenario or an exception. |
| If you drive through a red light at an intersection, you will get a ticket. | Omission of implicit action, such as 'the user hits enter' or 'the event times out', or 'the record is not found'<br>A better specification is "If you drive through a red light, and your car is photographed by a traffic camera or stopped by a police officer, you will get a ticket." |
| If the interest amount is greater than 100, send the customer the notice. | Aliasing of 'interest amount'. Is it interest owing, earned, paid or anticipated? |
| The system shall provide collection of PC configuration data for a mass release by the Delivery/Fulfilment team. | Subjective interpretation<br>- The team is named Delivery/Fulfilment.<br>- Some projects call the group a Delivery team, others call it a Fulfillment team.<br>- There are two teams and either team can do a mass release, so the slash means 'or'.<br>- Both groups jointly do the mass release, so the slash means 'and'. |