Bridging the Understanding Gap

David Gelperin
ClearSpecs Enterprises
Golden Valley, MN USA
Email: david@clearspecs.com

***Abstract*—Understanding is based on knowledge, but is different. Developers must understand customers and their needs. If customers don't understand what they want or developers don't understand what is needed, these understanding gaps must be bridged. Understanding gaps should be assessed before bridging. This paper defines levels of understanding and then describes approaches to estimating understanding, assessing understanding gaps, and selecting bridging tactics. An example of gap management is included.**

Keywords - understanding levels; understanding gaps; estimating understanding; assessing gaps; bridging tactics

I. INTRODUCTION

One can distinguish four categories of requirements [1]: (1) **Business requirements** – state the goals and needs of the enterprise (2) **Stakeholder requirements** [including user role requirements] – state needs of groups or individual stakeholders (3) **Solution requirements** – state characteristics of a solution (4) **Transition requirements** – state temporary capabilities needed to facilitate transition to a new system. This paper focuses on solution requirements.

Understanding (of requirements) is based on knowledge, but is different. Understanding enables reliable derivation and prediction. Developers must understand customers and their needs.

The challenge of requirements development is to "bridge understanding gaps". Sometimes customers don't understand what they want or developers don't understand what is needed.

Understanding gaps should be assessed before bridging. Perhaps the gap is small and bridging will be easy. Perhaps the gap is too large and the project is infeasible. Perhaps the gap is bridgeable, but the more we understand the gap, the more likely we will succeed in bridging.

To meet the challenge of assessing and bridging a gap, we view requirements development as analogous to orienteering. Orienteering is a sport/game that uses a map and compass/GPS (Global Positioning System) to navigate rapidly from a starting point to an ending point, sometimes across unfamiliar terrain.

We view requirements development as an activity/game that uses a map and a UPS (Understanding Positioning System) to navigate as rapidly as feasible from a starting requirements point to a sufficient understanding requirements point, sometimes across unfamiliar terrain.

II. LEVELS OF UNDERSTANDING

The primary goal of (solution) requirements development is to help developers acquire a sufficiently deep understanding of customer and user needs, via effective communication and cooperation, so they can do their jobs.

The following levels of understanding include two end points, no understanding and adequate understanding, and three intervals.

A. No Understanding

If developers or customers have little understanding of the application domain, then there will be no understanding or worse, misunderstanding of the requirements. For example, consider the requirement: Display atelectasis findings with highlighting. Looking in a medical dictionary we find that "atelectasis" means 'collapse of all or part of a lung'. Unfortunately no dictionary is sufficient for complete understanding of an application domain.

B. Unfamiliar Understanding

Requirements may be specified with familiar words used in unfamiliar ways. For example, a requirement might be: "Report each black box warning that is missing." While the words are familiar, understanding its detailed meaning requires an understanding of the language used to describe prescription drug ad compliance with FDA guidelines.

C. Familiar Understanding

This entails an understanding of some, of the fundamental entities, activities, relationships, and consequences in the application domain. For example, the investment rating companies had a limited understanding of mortgage backed securities prior to the recent financial crisis e.g. they didn't know enough about consequences.

D. Expert Understanding

This entails an understanding of most of the fundamental entities, activities, relationships, and consequences in the application domain. This does not entail knowing everything, but knowing almost all important things.

E. Adequate Understanding

This entails a sufficient understanding of all requirements needed to perform.

III. ESTIMATING UNDERSTANDING

You can promote developer and customer understanding by using "early acceptance test design" [3]. This entails the early creation and cross-functional group review of comprehensive acceptance test criteria and designs (preconditions, triggers, responses, post-conditions). Early design can be a powerful educational/communication tactic because such designs provide detailed, real-world examples of required behavior and their early review can be a powerful way to synchronize understanding.

You can use the results of early acceptance test design in a 'practice test' to estimate developer and customer understanding. Understanding can be estimated using (1) functional tests by asking for required responses to input situations and (2) quality tests by asking if 'behavior descriptions' satisfy requirements. Alternatively, developer practice tests can just be based on customer understanding.

In addition to test-based estimates, effective understanding estimation requires (1) a culture of open and safe communication that encourages truth telling and (2) the management of understanding estimates. For example, see the summary and tracking reports below.

Risk Factors

Risk Type	Essentials	Necessarys	Desirables	Supples
Superficial Dev Know	5	2	1	0
Limited Dev Know	17	4	0	6
Seriously Defective Reqts	0	0	0	
High Threat Level Reqts	0	0	0	
Reqts Issues	0	0	0	
Reqts Conflicts	0			

Figure 1. Risk Summary Report

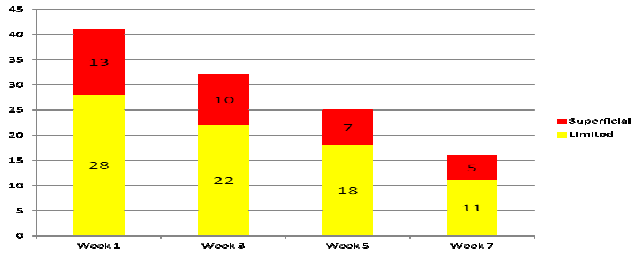

Figure 2. Understanding Tracking Diagram

Even thought boundaries between understanding levels are not clearly defined, the understanding level framework in Section 2 can be used safely, if boundary decisions are always resolved in favor of the level of lesser understanding.

If there is more than one developer, estimate the understanding of the developer that understands the least. If there is more than one customer, estimate the understanding of the customer that understands the most.

To make the following description clearer, we make the unrealistic assumption that both customer and developer understanding is homogeneous (i.e. each party understands all facets of the solution equally). Later, we remove this assumption.

IV. ASSESSING GAPS

We use the following reference map (grid) to "locate the position" of customer and developer understanding. The bottom and left sides (lines) represent adequate understanding. The top and right sides (lines) represent no understanding.

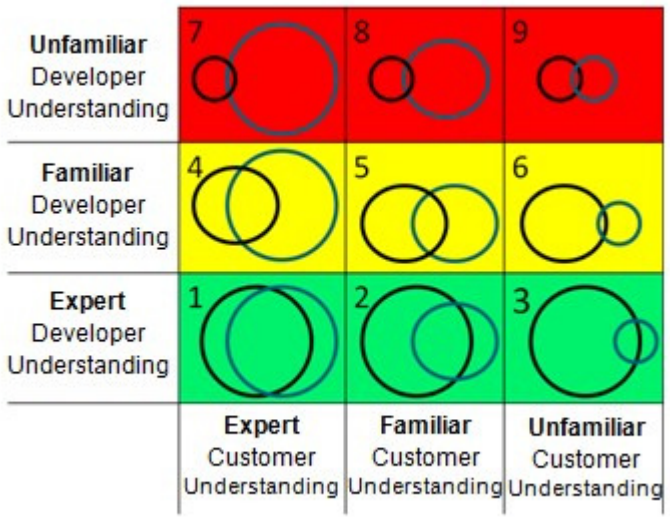

Figure 3. Understanding Map

At a moment in time, if the customer has an expert understanding of the requirements and the developer has an unfamiliar understanding, our Understanding Positioning System tells us that, at that moment, they are located in cell 7.

The goal is to move to the bottom as far and as fast as possible. If you start your project near the bottom, you have little requirements risk (i.e. a small gap) because the developers begin with an expert understanding. For example, when you need an e-commerce website, you select developers who have ten years experience developing such sites. Note that these developers have a deep understanding even though the requirements are not explicitly recorded.

V. SELECTING BRIDGING TACTICS

After locating your position on the understanding map, if you are not near the bottom, you must select tactics for "moving the developers down" i.e., increasing their understanding. Sometimes the path is hard either because customers understand little of what they need or they understand a lot, but don't agree. In some cases, the customers know exactly what they need, but the developers don't understand the language of the application domain. For example, to build a system for modeling organic molecules, some developers must have the equivalent of a Masters in Organic Chemistry to understand the requirements.

We select one or more of the following seven games to increase requirements understanding:

- **Bridge Bidding Conversations**, for mutual deeply understood requirements
- **10/20 Questions**, for requirements deeply understood by developers only
- **Jigsaw Puzzling**, for requirements deeply understood by customers only
- **Scavenger Hunt 1/2**, for requirements deeply understood by neither party
- **Negotiation**, for serious requirement disagreements among customers
- **Enculturation**, for use when developers do not understand the application domain
- **Decoding**, for use when developers do not understand an existing system that must be changed

Details about these games, except for Negotiation, are described in "Improve Requirements Understanding by Playing Cooperative Games" [5]**.** One form of Negotiation is described in [4].

Game selection is based on the following table:

Table 1. Game Selection Table

Cell	D U	C U	Primary Games
1	D	D	BB Conversations (with conventions)
2	D	L	20 Questions, BB Conversations
3	D	S	20 Questions
4	L	D	10 Questions, BB Conversations, Jigsaw Puzzling, Coached Enculturation, Decoding
5	L	L	10 Questions, BB Conversations, Jigsaw Puzzling, Scavenger Hunt 1/2, Negotiation, Coached Enculturation, Decoding
6	L	S	10 Questions, Scavenger Hunt 1/2, Decoding
7	S	D	Jigsaw Puzzling, Coached Enculturation, Decoding
8	S	L	Jigsaw Puzzling, Scavenger Hunt 1/2, Negotiation, Coached Enculturation, Decoding
9	S	S	Scavenger Hunt 1/2, Decoding

VI. HETEROGENOUS UNDERSTANDING

The discussion in the prior two sections (4 & 5) assumes that both customer and developer understanding is homogeneous (i.e. each party understands all facets of the solution equally). This is rarely True. The norm is to have different levels of understanding of different facets (capabilities or features) or parts of facets.

This means that developer or customer understanding of the requirements will rarely be mapped into a single cell and if it is, the cell is most likely to be 9, were neither party understands very much. Normally, understanding will map into multiple cells.

The more complex the mapping, the greater its value because seeing the mapping enables customers and developers to understand the diverse forms and occurrences of cooperation needed to be successful in discovering and communicating requirements.

VII. GAP MANAGEMENT EXAMPLE

Bob wants to create a website that helps duplicate bridge pairs [2] receive training in defensive play and bidding. Each training board (set of four hands) on the site will include an expert analysis that describes what each member of the pair should be thinking about and the actions they should take. These boards will differ from those you may have seen in newspapers or books, because they will contain at least one lesson for each member of the pair. The boards will be prepared by bridge authors/teachers. The website will provide training rooms (chat rooms) and support the selection and presentation of the training boards as well as the inclusion of a (human) coach. While sites exist where bridge pairs can play (practice), there is no site where they can be trained.

Bob's obvious strategy is to narrow the initial understanding gaps by finding skilled web developers, who are also duplicate bridge players. A three month search yields no results. Developers do not need to be expert bridge players because the bridge expertise will be contained in the boards.

However, to understand the requirement for a clean interface with a familiar look, developers need some experience playing duplicate and an understanding of potential player demographics. Most developers are two or three decades younger than many players. Duplicate experience will help developers understand why bidding sequences must use a close analogue of bidding boxes [2].

This example centers on cell 8 with the interim goal of moving to cell 5. Coached Enculturation [5] is at the core of the bridging strategy. Bob will take the developers to a duplicate club, where they will watch the play, talk to players after the game, and learn the rudiments of duplicate play. Bob will have the developers create a basic duplicate bridge glossary. Early acceptance test design will be used and developer understanding of each requirement will be tested. Developers will be challenged to identify new requirements based on their growing understanding. Understanding will be monitored and tracked. New tactics will be selected based on the outcomes of these activities.

The developer's deep understanding of site security will be useful. Their shallow understanding of site accessibility by hearing or sight impaired players will need to be addressed.

VIII. CONCLUSION

Bridging understanding gaps is of little concern if:

- Developers have a deep understanding of the key facets of the new product or change
- Bridging happens "naturally" during your current development process

Otherwise, failure to design and use an effective bridging strategy will results in:

- Best case – significantly increased project cost and duration
- Worst case – development failure, especially on unfamiliar projects

REFERENCES

[1] …, **A Guide to the Business Analysis Body of Knowledge** (BABOK Guide) Version 2.0 International Institute of Business Analysis 2009

[2] …, Wikipedia **http://en.wikipedia.org/wiki/Duplicate_bridge**

[3] Adzic, Gojko (2009) **Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing** Neuri Limited

[4] Boehm B., Grünbacher P. (2000), "Supporting Collaborative Requirements Negotiation: The EasyWinWin Approach", In: Landauer C., Bellman K.L. (eds.), **Proceedings International Conference on Virtual Worlds and Simulation**, San Diego, January 23–27.

[5] Gelperin, David (2011) "Improve Requirements Understanding by Playing Cooperative Games" **Proceedings of INCOSE International Symposium 2011**