

Boundary Objects

Brian Marick

marick@testing.com

www.testing.com, www.visibleworkings.com

Background, Definitions, Examples

An agile software project is a *community of interest* that draws its members from at least two different *communities of practice*.

A *community of practice* is a group of people who do a certain type of work, talk to each other about their work, and derive some measure of their identity from that work (refs). Programmers are a community of practice. Accountants are another.

A *community of interest* involves members of distinct communities of practice coming together to solve a particular problem of common concern (Arias and Fischer 2000). A team of programmers and accountants replacing a company's accounting system is a community of interest.

A community of interest can expect to face more communication problems than a community of practice. As Arias and Fischer (2000) write, "Fundamental challenges facing communities of interest are found in building a shared understanding of the task at hand (which often does not exist upfront, but is evolved incrementally and collaboratively...). Members of communities of interest need to learn to communicate with and learn from others who have a different perspective and perhaps a different vocabulary for describing their ideas. [They need to] establish a common ground and a shared understanding."

In agile projects, the community of interest always includes members of two communities of practice. There are the programmers, and there is one or more representative business experts (variously called "customers", "goal donors", and the like). Agile projects assume high-bandwidth, highly iterative, face-to-face conversation is the best way to guide a project.

But conversation about *what*? In agile projects, there are at least two topics of conversation, two semi-physical objects that programmers and business experts can gesture toward while talking. One is working software: an intermediate version of the final product whose operation the business expert can observe and judge. The other is some list of future tasks, such as XP story cards (ref) and Scrum backlogs (ref).

I believe these topics of conversation are what Star and Griesemer (1989) call *boundary objects*. Boundary objects have several important properties:

If x is a boundary object, people from different communities of practice can use it as what Chrisman (XXX) calls a COMMON POINT OF REFERENCE for conversations. They can all agree they're talking about x .

But the different people are not actually talking about the same thing. They attach DIFFERENT MEANINGS to x . For example, a story card that says "allow alpha chars in customer ID field" might be, to a programmer, a reminder to change class definitions

and update a database schema. To the business expert, it might represent an enabling step in merging the operations of two companies.

People use boundary objects as a MEANS OF COORDINATION AND ALIGNMENT (Fischer and Reaves 1995). Story cards are a tool XP projects use to align what the programmers build with what the business expert wants.

Despite different interpretations, boundary objects serve as a MEANS OF TRANSLATION. If it becomes important that the programmer understand more about business operations being merged, the story card can be used to smooth the process of explanation (for example, by delving more deeply into the meaning of the words on the card).

Boundary objects are PLASTIC enough to adapt to changing needs. And change they do, as communities of practice cooperate. Boundary objects are WORKING ARRANGEMENTS, adjusted as needed. They are not imposed by one community, nor by appeal to outside standards (Bowker and Star 1999).

The boundary object must satisfy DIFFERENT CONCERNS SIMULTANEOUSLY. In agile projects, the brief task descriptions and the conversation around them satisfies the business expert that something of actual business value will soon be produced while also satisfying the programmers that they are not committing to do more than they can.

Let me give two brief examples of boundary objects. Please note that the references give many more details and describe other boundary objects.

A museum's goal (Star and Griesemer 1989)

Star and Griesemer originally wrote about Berkeley's Museum of Vertebrate Biology in the period 1907-1939. Much as with the "allow alpha chars in a customer ID field" example, a goal was used as a boundary object. In this case, the goal might be stated as "preserve the natural fauna of California". To Grinnell (the curator), this goal was a means to the end of elaborating Darwinian theory – specifically, seeing how change in the environment drove natural selection. That end required the collection of a vast amount of detailed information about fauna and the environment they lived in. California was the boundary of his natural laboratory.

The museum's collection was enabled by conservationists who saw the flora and fauna of California disappearing and felt that it needed to be preserved while there was still time. They provided both funding (one, Annie Alexander, paid for the museum) and amateur collecting services. To them, the goal meant a quite different thing. However, Grinnell was able to use the boundary object to motivate them and guide their collecting – he could use it to explain things in their terms while using it for his own purposes (for example, to decide what data should be collected along with a specimen).

At the same time, Grinnell had to work with the University administration. To them, preserving the fauna of California had yet another meaning. It fit into their mandate of serving the people of California. It also fit into their goal of competing with elite eastern universities in terms of funding and prestige. The Berkeley Museum would be

of the same class as eastern museums, though of a different type because of its focus on covering a delimited region with better data. So the boundary object was suited to enlisting their support.

A drug company's committee (Frost et al, XXX)

Ivermectin is a popular drug for deworming animals. Onchocerciasis (river blindness) is a chronic illness that's a particular burden in sub-Saharan Africa. Since river blindness is caused by a worm susceptible to ivermectin, the manufacturer (Merck) desired to donate ivermectin to fight the disease. That presented some problems. For example, it would not be in Merck's interest if the bulk recipients responsible for redistributing ivermectin to people instead resold it into the lucrative veterinary market. On the other hand, it would also not be in Merck's interest to tell the recipients (including national governments that are markets for other Merck drugs) that they are not competent or trustworthy enough to receive ivermectin. Merck needed organizational distance.

The solution was for Merck to donate the drug to a non-profit non-governmental organization. An independent expert committee would make the decision about which applicants (both governments and non-governmental organizations) would then receive the drug. This committee is a boundary object. To Merck, it provides distance: Merck donates the drug, reaps the benefits in good will and tax deductions, but is insulated from political repercussions. To the bulk recipients, the committee is the dispassionate judge of applications, end-point of an application process, and advisor during implementation.

It's worth noting that ivermectin itself is a boundary object. To Merck, it is a drug to make individual patients better, as are most of its drugs. But other parties view it as a public health drug, because it interrupts transmission of the disease and can thus reduce its prevalence.

What we might talk about

In conventional projects, what are variously called functional tests, product tests, end-to-end tests, or acceptance tests are not good examples of boundary objects. I can argue that they are both a weak MEANS OF COORDINATION AND ALIGNMENT and also that they do not, in reality, SATISFY ANY CONCERNS of the programmers.

I think that agile acceptance tests in the "specification by example" style can do better. Let's talk about how. In particular, if we want acceptance tests to be good boundary objects, how does that affect our answers to these common questions:

Who should write acceptance tests? The customer? A tester acting as an avatar of the customer? The programmers? All of the above? (If so, how?)

What notation should be used in the acceptance tests? FIT-style tables? RoleModel Software style sugar-coated Ruby? My own preference for lighter sugaring of Ruby, combined with the assumption that customers never read tests alone? A mixture? Mixed how?