Mary Had A Little Lamb

Terry Yin

November 5, 2014

Every time I heard my daughter singing "Mary had a little lamb", I couldn't help but thinking "Mary should have some vegetable as well."

The "Mary had a little lamb" story is borrowed from Weinberg's book (Gause & Weinberg 1990, p.54).

There are two essential questions in making a software system, **WHAT** we are making, and **HOW** to make it. The **system analysis** phase is mostly about WHAT, since that's the most urgent question. We call it **requirement analysis**. But sometimes it's also about **HOW**, since in some situation, the feasibility and alternatives of the solution are also urgent question. In that case, we might also need to do operational research like the **feasibility study**.

If communication breakdowns during the system analysis, we will end up making the wrong software. So how can we avoid communication breakdown? Let's look at the reasons for communication breakdown respectively and see what we can do to improve it.

1 Ambiguity

As the "Mary had a little lamb" story told us, human language can be ambiguous. There are some approaches in modern software development to avoid it.

1.1 Specification By Example

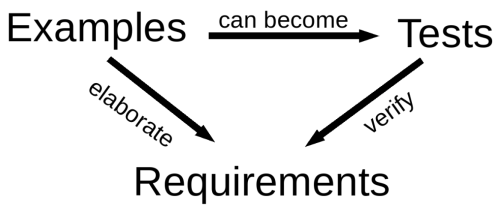
As the "Mary had a little lamb" story told us, human language can be ambiguous.

There's a simple but very powerful tool to avoid the ambiguity, that is the question "can you give me an example?"

The examples we use to clarify the requirement shouldn't be the general example, but need to be very specific scenarios, for instance, "given Craig is a VIP user, when he purchase the SbE book, he should have 50% discount." And if you feel one specific example is not general enough to cover the requirement, you can always add another specific example. And more.

Figure from *Bridging The Communication Gap* (Adzic 2009, pdf version p.27).

This is quite different than the traditional approach to analyzing the requirement. Traditional approach like use cases try to generalize the requirement and systematically cover the requirement. SbE doesn't aim to systematically cover the requirement but focus on value and priority of each scenario.



1.2 System Metaphor

Specification by Example is used to communicate the requirement. How shall we communicate the HOW part from the system analysis? One approach is called **system metaphor** Beck (2000). A system metaphor is a story that even the client can understand to explain how the system works. It helps to unify the terms we use across the whole development team.

2 Handoff

Being a Chinese myself, it surprised me to know that Chinese Whisper is a communication game played around the world, especially in UK [colleagues, confirmation needed here]. We have the same game in China but we have a different name for it. (And no, it's not called British whisper.) It's a hilarious game showing us that a piece of message can be totally distorted after being retold several times.

The traditional software development typically involves many moments of handoff. The business analyst passes the requirement to the architect. The architect passes the design decisions to the developer. The developer passes the software to tester. In each of these handoff, there will be information lost and mistakes.

2.1 Feature Team

One of the modern approaches to avoid handoff in software development is to build the **feature teams** Deemer et al. (2012). Feature team, in contrast to the traditional **single-disciplined team** or **component team**, is a small software development team includes all the necessary disciplines to deliver a software feature. One of the benefits of a feature team is to minimize the handoff.

2.2 Stop Writing Start Talking

The purpose of many documentation in traditional software development is to have the handoff. And that is usually not a good way of communication. Face-to-face communication is often more effective. **Specification workshop** is one of the collaborative ways to do system analysis (Adzic 2009, p.59). Instead of the requirement owner "push" the requirement to the development team, in the workshop, development team "pull" the information from the requirement owner, by asking questions like "can you give me an example . . . "

3 Long Feedback Cycle

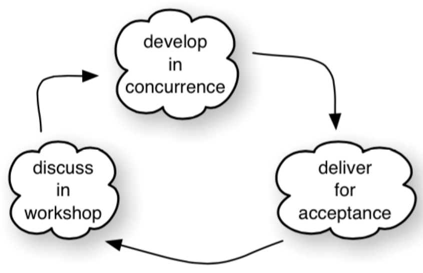
If our understanding about the requirement is wrong, the long feedback cycle will make it worse. We need shorter PDCA cycles to continuously verify our analysis, Wikipedia (2014).

3.1 Iterative development

The traditional waterfall approach has been proved not working and the modern software development model is always iterative, Larman & Basili (2003). Iterative development aims to get full feedback as early as possible. The full feedback should not be from any interim product, but the real working software.

3.2 Acceptance-Test Driven Development

An even more detailed approach for system analysis is called Acceptance-Test Driven Development (Larman & Vodde 2010, ATDD, p.42 59). This is highly related to Specification by Example as mentioned earlier. The requirement is explored in a collaborative way in a workshop. During the iteration, the acceptance test and the software are developed in concurrence. At the end of the iteration, the acceptance test can be used to confirm the requirement has been fulfilled.



4 Conclusion

Communication can breakdown. The breakdown of communication during system analysis is very costly. We need to avoid the ambiguity, handoff and long feedback cycle.

This paper is mainly based on the work of Craig Larman, Bas Vodde and Gojko Adzic. If you are interested in this topic, I would like to recommend the 4 books in the references written by them.

References

Adzic, G. (2009), *Bridging the communication gap: specification by example and agile acceptance testing*, Lulu. com.

Beck, K. (2000), *Extreme programming explained: embrace change*, Addison-Wesley Professional.

Deemer, P., Benefield, G., Larman, C. & Vodde, B. (2012), 'The scrum primer 2.0'.

Gause, D. C. & Weinberg, G. M. (1990), 'Are your lights on', *Dorset House* .

Larman, C. & Basili, V. R. (2003), 'Iterative and incremental development: A brief history', *Computer* **36**(6), 47–56.

Larman, C. & Vodde, B. (2010), *Practices for scaling lean & agile development: large, multisite, and offshore product development with large-scale Scrum*, Pearson Education.

Wikipedia (2014), 'Pdca — wikipedia, the free encyclopedia'. [Online; accessed 5-November-2014].
URL: *http://en.wikipedia.org/w/index.php?title=PDCA&oldid=630479736*