What Language Does Agile Speak?

authors

affiliation,
address
{Email}
url

Abstract. *Collaboration-intensive Agile practices are dependent on the development team understanding the customer's perspective and requirements. Through a Grounded Theory study of Agile teams in New Zealand and India, we discovered that a gap between the teams' technical language and the customers' business language poses a threat to effective team-customer collaboration. We describe this language gap and the 'Translator' role that emerges to bridge it.*

Key words: Agile Software Development, Customer Collaboration, Language Gap, Translator, Grounded Theory

1 Introduction

Agile software development requires regular collaboration between development teams and their customers, in an effort to build software solutions that meet the customers' needs [2, 9, 17]. Collaboration-intensive Agile practices, such as release and iteration planning, user acceptance testing, and reviews; are unlikely to succeed unless development teams are able to effectively understand the customers' prespectives and requirements [6, 12, 13]. Mistranslation of customer requirements can result in development of late and unreliable software systems [8]. Through a Grounded Theory study, we discovered this gap between the teams' *technical* language and the customers' *business* language threatens effective team-customer collaboration. In this paper, we describe this language gap and the *Translator* role that emerges to bridge it.

2 Research Context and Method

We used Grounded Theory [5] as our research method as it enables the study of social interactions and behaviour — a focus of Agile software development. We interviewed 20 Agile practitioners across 12 different software organizations in New Zealand and India using Scrum and XP over a period of 2 years. Data was collected through face-to-face, semi-structured interviews using open-ended questions. The interviews were approximately an hour long and focused on the challenges faced on Agile projects and the strategies used to overcome them. Open coding was used to analyze the interview transcripts in detail [4]. Using GT's *constant comparison method* data from interview transcripts was raised in levels of abstraction from key points to codes to concepts to

categories [4, 5, 17]. In the following sections, we describe the *Language Gap* and the *Translator* role that emerged from data analysis. We have selected quotations drawn from our interviews that serve to highlight the results and that are spread across most participants. Figure 1 shows the participant and project details.

Partipants	Agile Position	Agile Method	Org. Size	Country	Domain	Team Size	Project Duration	Iteration
P1	SM	Scrum & XP	S	NZ	E-commerce	4	2	4
P2	Cust Rep	Scrum	XS	NZ	Entertainment	6 to 8	9	4
P3	AC	Scrum & XP	S	NZ	Government Education	4 to 9	4	2
P4	AC	Scrum & XP	XL	NZ	Telecom & Transportation	6 to 15	12	4
P5-P7	Dev × 2, AC	Scrum & XP	S	NZ	Environment	4 to 6	12	1
P8-P10	Dev × 2 , BA	Scrum	M	NZ	Health	7	9	2
P11-13	SM, Dev, Tester	Scrum & XP	S	India	Software Development & Consultancy	5	6	2
P14	AC	Scrum & XP	L	India	Telecom	8 to 15	3	4
P15	Designer	Scrum & XP	S	India	Web-based services	5	1	2
P16, P17	AC × 2	Scrum & XP	M	India	Software Development	7 to 8	3 to 6	2
P18	AT	Scrum	XS	India	Agile Training	7	8	3
P19, P20	Dev, HRM	Scrum & XP	M	India	Software Development	15	12	1

Fig. 1. Participant and Project Contexts. (Agile Position: Agile Coach (AC), Developer (Dev), Customer Rep (Cust Rep), Business Analyst (BA), Senior Management (SM), Agile Trainer (AT), Human Resource Manager (HRM); Organizational Size: XS $<$ 50, S $<$ 500, M $<$ 5000, L $<$ 50,000, XL $>$ 100,000 employees; Project duration is in months & iterations are in weeks.)

3 Results

3.1 The Language Gap

As a result our GT analysis, we discovered that development teams and their customers use different languages when collaborating on Agile projects. While the development teams use a more *technical* language composed of technical terminology, their customers are used to a more *business* langauge composed of jargons from the customers' business domains.

> "*The terminology that developers are using and customers are using, is very different.*" — P5, Developer, NZ

> "*We have meetings with [the customers] and obviously there are some gaps in the language and in the jargon.*" — P6, Developer, NZ

The language gap between development teams and their customers poses a threat to effective team-customer collaboration by limiting their understanding of each other's perspectives. The *technical* language used by development teams was difficult for their customers to understand:

> "*I might explain something in a very cryptic, technological way and [the customers] won't understand a word!*" — P8, Developer, NZ

"*If you give people information with a technical bent, and they're business people, they switch off.*" — P3, Senior Agile Coach, NZ

Similarly, the customers' *business* language was difficult for the development teams to understand, as a Scrum Product Owner (customer) noted:

"*They are very smart developers and they are really into 'yes we can code this or make this thing', but not really putting themselves in the user's shoes or the client's shoes.*" — P2, Product Owner (customer), NZ

Translation between the two languages was found to be necessary in order to promote understanding between the two language groups and maximize the effectiveness of collaboration-intensive Agile practices. The need for translation led to the emergence of an informal *Translator* role on these Agile projects.

3.2 Translators — Bridging the Gap

The *Translator* is a development team member responsible for facilitating communication and collaboration between the team and their customers by translating between their respective languages:

"*We'd need [a translator] anyway to translate [my technical terminologies], to make it mean something [in the customer's language] (laughs).*" — P8, Developer, NZ

Professional Analysts Initial analysis of some interviews revealed the role of the *Translator* was best suited to Business Analysts (P2, P3, P8-P10). Business Analysts were considered suitable candidates for the *Translator* role because of their perceived ability to understand both *technical* and *business* languages and to act as a bridge between the two.

"*Some things fell through the cracks because we didn't have a good BA...A BA from the beginning would have been really useful.*" — P2, Scrum Product Owner (customer), NZ

"*When we go into the meeting to talk to business [customers], we try to be non-technical...we work with BA a lot...it pays to have that BA [in the meeting].*" — P9, Developer, NZ

This observation was confirmed by the Business Analyst identified as the *Translator*, on the same team:

"*I was able to put across the policies and the procedures that the business service centre needed to be portrayed along a technical background.*" — P10, Business Analyst, NZ

Natural Communicators As our research progressed from studying relatively new Agile teams (learning Agile practices) to more mature teams (fluent in use of Agile practices, for usually more than a year), it became evident that the *Translator* role was not limited to be professional analysts and could be potentially played by anyone on the team with good communication skills and understanding of business concerns.

> "*...translators...understand the concerns of the business and translate them into priority elements that the development group can actually focus on to achieve...Somebody who wants to do it, who has this compulsion 'let me translate, let me help'...sometimes a PM, sometimes it's a BA, sometimes it's a developer, a tester.*" — P4, Senior Agile Coach, NZ

We found most members of mature Agile teams were bilingual: all able to play the *Translator* role. Some participants ensured that they were "*hiring smart, pragmatic communicators*" (P7) with innate *Translator* skills when recruiting for an Agile team.

> "*strong public-oriented skills...to solve the business problem of the customer...more important to understand the customer and their requirements...you have to be very smart enough to get the requirements [and] understand the business intent when you solve a problem.*" – P20, Human Resource Manager, India

3.3 Becoming a Translator

Several participants believed that certain tools and techniques could be used to become a successful *Translator*:

Using a Dictionary We found one of the Indian teams using a '*dictionary*' to assist everyone on the team become a *Translator*. This *dictionary* was an online editable document (wiki) populated by the customers with business terms, their meaning, and their context of use. These business terms were translated directly into code by the team using the same variable names, providing one-to-one mapping between the customers' *business* terms and their *technical* implementation for a given project. The customers were able to view and edit the contents of the evolving *dictionary*.

> "*we have extensive documentation...a wiki [where the customers] have explained their whole infrastructure...as and when they build up the requirements they come and edit the document...its kind of like a glossary and also the rules that figure in that world of theirs...we capture all that and ensure our domain is represented exactly like that in code.. ..so when they say 'a port has to be in a cabinet which has to sit in a rack' it directly translates to code!*" — P19, Developer, India

Using Story Cards User stories are written down by on story cards by customer representatives in the *business* language with domain specific requirements and are broken down into development tasks by the team. The tasks are written in *technical* language that are specific enough for development to commence. The actual translation of business requirements into technical tasks happens when user stories are broken down into technical tasks, which can be a challenge:

> "*The biggest issues with the development team...the translation of what the client wants into something the development create. So you have a story card with some features on....how to turn that story card into part of a website?*" — P1, Senior Management, NZ

We found two stratgies used to overcome this problem (a) assigning a coach to guide new teams on the use of story cards (P3) till they got efficient in translating them into technical tasks and (b) frequent release of software to ascertain the correctness of the translation through customer feedback (P1).

Using Iterative Reasoning Another key to becoming a *Translator* was found to be the power of questionning proposed technical solutions repeatedly till the abstract business reasoning behind the technical details was unraveled.

> "*why do we need that database back up procedure? or...database recovery? and it's right down at the technical level [asking] the question why, why, why till...you'll eventually discover there's a good business reason for having it.*" — P3, Senior Agile Coach, NZ

Using iterative reasoning, techincal soltuions could be abstracted to higher levels till they were clearly aligned with their business drivers.

Promoting Cross-functionality Interactions between members from diverse disciplines fosters understanding of the project from multiple perspectives [11]. As the team learns to understand their customer's perspective, they achieve greater levels of cross-functionality and are able to translate between their respective languages. An experienced Agile coach disclosed that the secret to acquiring the *Translator* skills was the quality of cross-functionality.

> "*The whole thing with Agile is getting people to be more cross-disciplinary, to take an interest in somebody else's perspective...The moment you understand that cross-concern, you're teaching everybody to become a translator.*" — P4, Senior Agile Coach, NZ

4 Related Work

A proposed solution to the linguistic divide is the use of a ubiquitous (shared) language in diagrams, writing, and speech, based on a domain model [3]. We did not find it being used by our participants. We did, however, find that the *dictionary* used by our participants was conceptually similar to the data dictionary [7] but proved additionally valuable as an effective means of translating customer domain terms directly into code. Other studies have asserted the effectiveness of user stories in understanding customer requirements [10, 14, 15]. Our study suggests that story cards can be augmented by the *dictionary* to capture and translate customer requirements. Some studies have described individuals supporting customers by translating *technical* langauge to *business* language [10, 12]. In contrast, our *Translator* role was able to achieve the opposite for

the development team by translating *business* into *technical* language. Other practices such as metaphor [1] — aimed at increasing shared understanding of customer requirements — was not employed as a tool to overcome the language gap possibly because of its percieved difficulty of use [16].

5 Conclusion

We discovered that relatively new Agile teams often have one or two individuals playing the *Translator* role. In contrast, most members of mature Agile teams are bilingual — speaking *technical* language in development circles and translating *business* language when collaborating with customers. The skills of a *Translator* can be an attribute of professional training (business analysts), natural abilities (natural communicators) or can be acquired using existing Agile practices such as *story cards* and *cross-functionality* and adapted practices such as a *dictionary* and *iterative reasoning*. Our findings have direct implications for improving team-customer collaboration on Agile projects.

Acknowledgments Our thanks to all participating Agile practitioners. This research is generously supported by (blinded for review) grant and a (blinded) scholarship.

References

1. Beck, K. et al.: Extreme Programming Explained, 2nd Edition, Addison-Wesley, USA (2005)
2. Dybå, T., Dingsoyr, T.: Empirical Studies of Agile Software Development: A Systematic Review. J. Inf. Softw. Technol., 50(9-10), 833–859 (2008)
3. Evans, E.: Domain-Driven Design: Tacking Complexity in the Heart of Software (2003)
4. Georgieva, S., Allan, G.: Best Practices in Project Management Through a Grounded Theory Lens. E. J. Business Research Methods (2008)
5. Glaser, B., Strauss, A.L.: The Discovery of Grounded Theory. Aldine, Chicago (1967)
6. Grisham, P. S., Perry, D. E.: Customer relationships and Extreme Programming. In Proceedings of HSSE '05, ACM, New York (2005)
7. IBM: IBM Dictionary of Computing. 10th. McGraw-Hill, Inc. (1993)
8. Korkala, M., Abrahamsson, P., and Kyllonen, P.: A Case Study on the Impact of Customer Communication on Defects in Agile Software Development. In AGILE2006, USA (2006)
9. Highsmith, J., Fowler, M.: The Agile Manifesto. Soft. Dev. Magazine. 9(8), pp. 29–30 (2001)
10. Mann, C. and Maurer, F.: A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction. In ADC IEEE Computer Society, USA, PP.70–79 (2005)
11. Takeuchi, H. et al.: The new new product development game. Harvard Bus. Review (1986)
12. Martin, A., Biddle, R., and Noble, J.: The XP customer role: A Grounded Theory. In Agile2009. IEEE Computer Society. Chicago (2009)
13. Misra, S. C., Kumar, V., and Kumar, U.: Identifying some important success factors in adopting agile software development practices. J. Syst. Softw. 82, 11, pp 1869—1890 (Nov. 2009).
14. Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., and Still, P.: The impact of agile practices on communication in software development. J. Empirical Softw. Engg (2008)
15. Sharp, H. et al.: Collaboration and co-ordination in mature XP teams. Int. J. Hum.-Comput. Stud. 66, pp.506–518 (2008)
16. Sfetsos, P. et al.: Investigating the extreme programming system—An empirical study. Empirical Softw. Engg. 11(2), pp.269–301 (2006)
17. Blinded for review