Proceedings of the e-Skills for Knowledge Production and Innovation Conference 2014

Cite as: Mbekela, U., & Brown, I. (2014). Factors that influence misalignment between developers and testers in agile organizations, and alleviation strategies employed. *Proceedings of the e-Skills for Knowledge Production and Innovation Conference 2014, Cape Town, South Africa,* 203-210. Retrieved from http://proceedings.e-skillsconference.org/2014/e-skills203-210Mbekela817.pdf

Factors that Influence Misalignment between Developers and Testers in Agile Organizations, and Alleviation Strategies Employed

Unathi Mbekela and Irwin Brown
University of Cape Town, Cape Town, South Africa

mbkuna002@myuct.ac.za; irwin.brown@uct.ac.za

Abstract

The concept of alignment has been addressed in the context of business and information technology (IT) domains within organizations, but very little research has investigated the alignment of roles within units in an organization. Within the IT domain, research shows evidence of misalignment between the role of the software tester and the software developer in software development teams, specifically in agile organizations that adopt agile methodologies. Much research on this topic focuses on managing the problem and preventing the problem, but not alleviating it, because there is insufficient literature that investigates the factors that influence misalignment. The purpose of this paper therefore is to identify through a comprehensive literature review the factors that lead to misalignment between developers and testers, and to hence recommend strategies to alleviate the problem of misalignment.

Keywords: Misalignment, influence factors, developers, testers, agile organization, alleviation strategies

Introduction and Problem Statement

The relationship between developers and testers in information technology (IT) organizations is a subject of concern because studies have revealed that there is poor collaboration between the two parties (Dhaliwal, Onita, Poston, & Zhang, 2011). Moreover, developers and testers have very conflicting roles despite the fact that they are expected to work closely within their subunits (Zhang, Stafford, Dhaliwal, Gillenson, & Moeller, 2014). It is important for developers and testers to have a good working relationship because the work they do is critical to the success of agile organizations – i.e. IT organizations that employ agile software development methodologies. It is important to ensure that there is collaboration, cohesiveness and communication between these two parties (Cohn, Leffingwell, Larman, & Vodde, 2013). Davis (2014) suggests that managers of software development teams should emphasize the importance of teamwork and reiterate that every member of the software development team has an important role to play in ensuring the quality of the end product. The aim of the study is to investigate the factors that influence the misalignment between developers and testers in agile

organizations and subsequently explore the strategies that can be implemented to alleviate this problem. Prior research has mainly focused on managing the problem and preventing the problem but not alleviating it (Zhang, Stafford, Dhaliwal, Gillenson, & Moeller, 2014). The study discusses how alignment between developers and testers is articulated (Onita and Dhaliwal, 2011).The study then explores various underlying factors that lead to conflict between developers and testers, and suggests alleviation strategies.

Agile Methodologies and SCRUM Roles

Contrary to the traditional-plan driven organizations, agile organizations are concerned about people and collaboration rather than procedures and tools; software that works rather than extensive documentation; building customer relationships rather than negotiating contracts; and being change-driven rather than following a rigid plan (Wilson, 2013). Agile methodologies are defined as lightweight software development methods that have been introduced as a counter to the failures experienced with traditional software development methods (Popli & Chauhan, 2013).

The agile methodology is composed of various methodologies that can be used by software development organizations to effectively manage their projects, such as eXtreme Programming (XP) (Beck, eXtreme Programming Explained, 2000), SCRUM (Lyssa, 2011), and Dynamic System Development Methodology (DSDM) (Stapleton, 1995). The focus of the current research will be on the SCRUM methodology.

The SCRUM methodology suggests the creation of SCRUM teams composed of the following roles: 1) Scrum Master – the scrum master is responsible for removing any impediments or obstacles that may affect the successful completion of work in the team. 2) Product Owner – the product owner is responsible for communicating all the requirements of the project work as expected by the customers and for communicating the time lines associated with implementing the work to the team. 3) Team members – these are self-organizing teams from various functional units of an organization which are responsible for the development and testing of the work. Usually this group of people should be between 4 and 9 individuals in order to maximize productivity without jeopardizing the quality of the product. The developer and tester would fall in as team members (Schwaber & Sutherland, 2011).

Role of Developers and Testers in SCRUM teams

Software developers are usually involved in the software development lifecycle (SDLC) from the planning until the maintenance phase, because they initially meet with the customers or end users of the software they intend to develop. In these meetings they discuss the requirements of the customer and whether or not these requirements are feasible. Thereafter all the requirements are documented and forwarded to the programmers who in turn write the code. The code is then tested by software testers and if any bugs or issues arise, these are sent back to the programmers. This is done until the software developer is satisfied with the software, which they then demonstrate to the customer. If the customer is satisfied, the product is officially released (Bureau of Labor Statistics, 2014).

Software testers evaluate the quality of the software that has been developed by software developers to ensure that it meets the requirements as specified in the requirements specification document. Software testers usually join the SDLC at the implementation stage, once all the development has taken place and the software is ready to be tested. Software testers are responsible for ensuring the overall quality of the software has been met, and prevent software to be released if it is not free of bugs, errors and other problems (Bureau of Labor Statistics, 2014).

Conflict often arises between developers and testers in SCRUM teams. This conflict arises as a result of the conflicting roles, objectives and skill sets of the developers and the testers. The de-

veloper is responsible for designing the software as specified in the requirements specification, developing the code for the software and making sure that the software does what it has been set out to do; the testers' responsibilities involve identifying flaws, bugs or problems in the software that the developers have produced and overseeing the quality of the end product (Zhang, Stafford, Dhaliwal, Gillenson, & Moeller, 2014). The developer aims to design and develop working software efficiently while maximizing resources and minimizing the time used for the whole development process. Testers on the other hand strive to ensure that the software produced does not only work, but that it is of acceptable quality and effectively does what it has been set out to do (Zhang et al., 2014).

Misalignment in Software Development and Testing

Prior to discussing misalignment in the software development context it is worthwhile to investigate the various terms which also refer to alignment. Ullah & Lai (2013) refer to alignment as the fit between the business and IT strategy as well as a fit between the business structure and the IT structure. Chan & Reich (2007) refer to alignment as a link, thus defining alignment as the link between business plans and objectives and IT plans and objectives. Chan & Reich (2007) also refer to alignment as the degree of congruence between IT strategy and the business strategy. All these references to alignment conceptualise it as a state between business and IT. Baljit (2013) refers to alignment as the process of improving communication between the strategic decision makers of a business and the IT decision makers in order to bridge the gap that is created by differing roles, goals and values between the IT managers and executive managers. Misalignment on the contrary would be defined as the opposite of alignment – i.e., the breakdown in communication between IT decision makers and business decision makers. This misalignment concept can be applied more specifically to the case of the breakdown in communication that occurs between developers and testers.

Onita and Dhaliwal (2011) propose a developer-tester alignment (DTA) model to explain the importance of alignment between developers and testers. This model draws from notions that are extracted from the business-IT alignment model and thus the DTA model defines alignment as follows: "Alignment between development and testing strategy/capabilities can be defined as the strategic and operational fit between a firm's development and testing functions" (Onita & Dhaliwal, 2011, p. 50). The DTA model focuses on establishing some form of common ground between developers and testers because both parties cannot succeed in achieving the objectives of the IT function if they do not work as a team and support each other in doing their work. Figure 1 illustrates the DTA model as discussed by Onita and Dhaliwal (2011):

The DTA model in Figure 1 proposes that the development and testing strategies (Q1 and Q2) need to be aligned in order for the two subunits to successfully meet the objectives of their subunits. If the strategies between the two subunits are not aligned, this may result in a deviation from the direction that should be taken to reach the goal. The arrow numbered 1 shows the alignment of the two strategies. Secondly, the model suggests that the development capabilities and the testing capabililities (Q3 and Q4) should also be aligned in order for the subunits to work effectively together, otherwise the mismatch in capabilities may cause a breakdown in their work relationship and adversely affect the goal. The arrow numbered 2 shows the alignment of the capabilities. Thirdly, the model recommends that there should be an alignment in how the development division executes their strategies and capabilities. In order to implement the processes, skills and architecture for development, the relevant scope, governance and resources must be in place to support the implementation of the above capabilities otherwise there will be a misalignment between these two concepts (Onita & Dhaliwal, 2011).

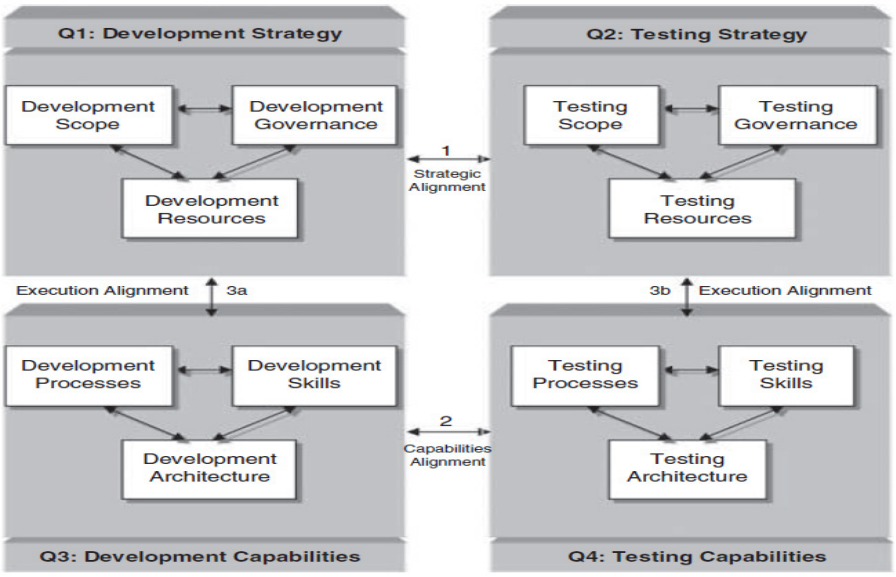

Figure 1. Alignment Model for Development and Testing
Source: Onita & Dhaliwal (2011)

Factors that Influence Misalignment between Developers and Testers

In order for the developer and the tester to collaborate and communicate effectively, they need to be placed in close proximity in the agile organization which means that they must be placed in one SCRUM team (Cohn et al., 2013). Having a developer and a tester in one SCRUM team has raised the argument of whether a tester is necessary or not in the team because developers do somewhat test their work before it is officially released to production environments (Sumrell, 2007). The need has been identified for testers to form part of a SCRUM team because the testing that the developers do relates to the units of the code that they have developed and not to the overall finished product - code may be correct, but that does not mean that the overall functionality works according to the specification (Sumrell, 2007). Factors identified as contributing to misalignment will be discussed next.

Lack of Communication

Lack of communication between developers and testers in project teams plays a big role in contributing to the conflict between these two parties. Organizations should identify strategies that will create an environment where communication, knowledge sharing and collaboration can be easy between these two parties (Zhang et al., 2014). Since both developers and testers play an important role in ensuring that the quality of the software product is satisfactory, there ought to be a sense of togetherness and unity in their work ethic. Miscommunication between them has adverse effects on achieving the strategic objectives of the IT unit (Ammann & Offutt, 2008).

Zhang et al. (2014) propose a three-layer conflict model which depicts the factors that influence the conflict between developers and testers. The three-layer conflict model is illustrated in Figure 2.

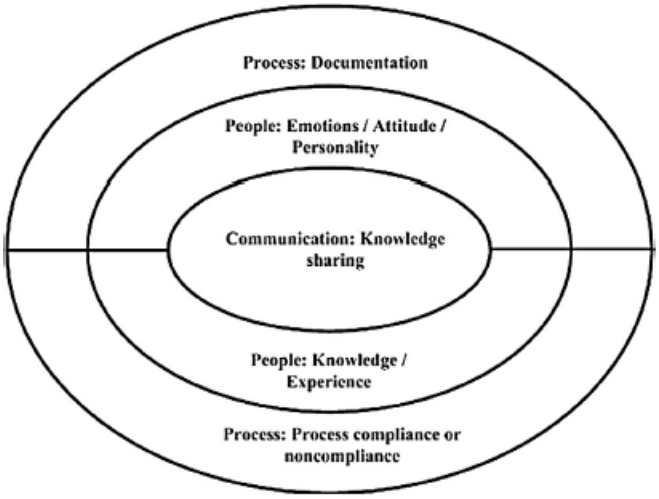

Figure 2. Three-Layer Conflict Model
Source: Zhang et al. (2014)

The model infers that the source of conflict stems from three components: process, people and lack of communication, but centrally lack of communication.

Poor Collaboration

Marczak & Damian (2011) emphasize the importance of collaboration, coordination and communication specifically in a software development team to ensure the success of a project.Poor collaboration between those involved in performing the core tasks in the software development team could result in a failure of the project. Hence poor collaboration between developers and testers contributes to misalignment (Ammann & Offutt, 2008).

Process Non-Compliance

The three-layer conflict model of Zhang et al. (2014) suggests that developers ought to be efficient in updating bug tracking tools and in informing the testers of changes or updates to code. Developers often struggle to comply with the processes set out for the team. Non-compliance with process contributes to misalignment.

Differing Mindsets

The mindset of a developer or tester will contribute immensely to the conflict between these two parties. This is due to the fact that the developer or tester will join a team with one or more of the following attitudes towards the other party: "1) You don't know much about what I do so I'll keep you in the dark or disregard you until you just go away, 2) I don't like the fact that you are in this team. I will strive to keep you off my back as far as possible, 3) I know there is a good reason for you being in this team and that I should be haste in my work to let you catch up but I will not worry myself about you, 4) I don't like the way you code, Your testing is useless" (Lagestee, 2013). These attitudes impact the relationship between developers and testers in a negative way and should be eliminated before they cause further harm to the team structure (Lagestee, 2013).

Blaming

The level of experience, knowledge and skills of the developer and the tester may not always be aligned. This can result in a conflict between the two parties because the way the two parties will

understand and apply the work will be determined by these factors (Zhang et al., 2013). It is important for the two parties to be aware of such differences and find ways to work around them. Due to the fact that software testers do the final testing before a product is released to the production environment, they are often blamed when bugs or issues are found in the production environment and the developers are usually safe from the blame (Lagestee, 2013).

Strategies to Alleviate Misalignment between Developers and Testers

Table 1 illustrates strategies that can be employed to alleviate the sources of misalignment between developers and testers. These include regular meetings (Marczak & Damian, 2011); creating an environment to facilitate communication (Zhang et al., 2014); developing a shared understanding (Jasbir et al., 2011), and both parties agreeing to take full responsibility (Lagestee, 2013).

Table 1. Factors Influencing Conflict between Developers and Testers and Strategies to alleviate them

No.	Factors influencing misalignment	Reference/s	Strategy to alleviate factor	Reference/s
1	Lack of communication/knowledge sharing between developers and testers.	Tripathi & Kumar Goyal (2014)	Physical gathering - weekly or daily meetings.	Marczak & Damian (2011)
2	Poor collaboration between the two parties.	Marczak & Damian (2011)	Physical gathering - weekly or daily meetings.	Marczak & Damian (2011)
3	Non-compliance to process usually by developers.	Zhang, Stafford, Dhaliwal, Gillenson, & Moeller (2014)	Create an environment that facilitates effective and efficient communication.	Zhang et al. (2014)
4	Different mindsets of developers and testers.	Lagestee (2013)	Shared understanding of objectives and good match of skills.	Jasbir, Colin, Robin, & Xihui (2011)
5	Blaming one another when quality of software is poor.	Zhang, Dhaliwal, Gillenson, & Stafford, 2013 and Lagestee, (2013)	Shared understanding of objectives and good match of skills. Both parties must agree to take full responsibility for the qaulity of the end product,	Jasbir, Colin, Robin, & Xihui (2011) Lagestee (2013)

Conclusion

From the literature review, the author identified that prior empirical research that has addressed the conflict or misalignment between developers and testers is not very context specific. Some of the literature addresses this problem in traditional-plan driven organizations whilst the literature that does refer to agile organizations does not specify that the developers and testers must be in the same location, part of the same team and must be expected to communicate and collaborate effectively. Additionally, the literature that discusses misalignment within agile organizations does not refer to agile organizations that are using one or more of the agile methodologies to manage their software development projects. The literature also reveals that most studies that propose strategies to address this problem do not come up with preventative measures but rather with measures that can manage the problem. It is evident that this approach has not been very successful because current research reveals that the factors which influenced misalignment between developers and testers a decade ago are still prevalent nowadays. Further research needs to be done in agile organizations to investigate why the misalignment between developers and test-

ers is still prevalent, but this research should be more context specific rather than general and the research should aim to identify how to alleviate the factors that cause this misalignment.

References

Ammann, P., & Offutt, J. (2008). *Introduction to software testing.* New York: Cambridge University Press.

Baljit, S. (2013). Demystifying business IT alignment. *International Journal of Science & Technology, 3*(1), 20-26.

Beck, K. (2000). *eXtreme programming explained.* Boston: Addison-Wesley.

Bureau of Labor Statistics. (2014, January 08). *Software developerss.* Retrieved from U.S. Department of Labor: http://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm

Chan, Y., & Reich, B. (2007). State of the art: What have we learned? *Journal of Information and Technology, 22*(4), 297–315. doi:10.1057/palgrave.jit.2000109

Cohn, M., Leffingwell, D., Larman, C., & Vodde, B. (2013, January 15). *Scaled agile framework.* Retrieved from Scaled Agile Framework: http://scaledagileframework.com/developers-and-testers/

Davis, R. (2014). *How do you improve the relationship between developers and testers?* Retrieved from Rob Davis PE: http://www.robdavispe.com/free5/software-qa-testing-test-tester-2294.html

Dhaliwal, J., Onita, G. C., Poston, R., & Zhang, X. P. (2011). Alignment within the software development unit: Assessing structural and relational dimensions between developers and testers. *Journal of Strategic Information Systems, 20*(4), 323-342.

Lagestee, L. (2013, May 07). *Illustrated agile.* Retrieved from Illustrated Agile: http://illustratedagile.com/2013/05/07/when-developers-and-testers-collide/

Marczak, S., & Damian, D. (2011). How interaction between roles shapes the communication structure in requirements-driven collaboration. *IEEE 19th International Requirements Engineering Conference*, (pp. 47-56). Canada.

Onita, C., & Dhaliwal, J. (2011). Alignment within the corporate IT unit: An analysis of software testing and development. *European Journal of Information Systems*, *20*(1), 48-68.

Popli, R., & Chauhan, N. (2013). Agile software development. *International Journal of Computer Science and Communication, 4*(2), 153-156.

Schwaber, K., & Sutherland, J. (2011, October). *Scrum organization*. Retrieved from Scrum Org: https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf

Stapleton, J. (1995). *DSDM – Dynamic system development method.* Boston: Addison-Wesley.

Sumrell, M. (2007). From waterfall to agile – How does a QA team transition? *IEEE Computer Society*, 291-295.

Ullah, A., & Lai, R. (2013). A systematic review of business and information technology alignment. *ACM Transactions on Management Information Systems, 4*(1), 1-30.

Wilson, C. (2013, July 21). *Matincor Inc.* Retrieved from Matincor Inc.: http://www.matincor.com/Documents/Plan%20vs%20Agile.pdf

Zhang, X., Dhaliwal, J., Gillenson, M., & Stafford, T. (2013). The impact of conflict judgment between developers and testers in software development. *Journal of Database Management, 24*(4), 26-50.

Zhang, X., Stafford, T. F., Dhaliwal, J. S., Gillenson, M. L., & Moeller, G. (2014). Sources of conflict between developers and testers in software testing. *Information and Management, 51*, 13-26.

Biographies

Unathi Mbekela is a part-time Masters student at the University of Cape Town. She graduated with an Honors degree at the University of Fort Hare in May 2009. She is currently pursing her Masters in Information Systems, with research intersts in Agile Software Development.

.\

Irwin Brown is a Professor in the Department of Information Systems (IS). He graduated with a PhD (UCT) in 2005. Irwin's research interests relate to the use of grounded theory methodology in IS studies, and issues around IS in developing countries contexts.