Problems and disadvantages of agile software development

Technology, Information Technology

Limitations of Agile Software Development According to Dybå and Dingsøyr (2008), creation of agile is as a result of reaction towards the then predominant but currently infamous waterfall model. Agile methodologies are applicable in place of traditional methodologies owing to failures experienced by projects using traditional methodologies. However, there is lack of detailed evidence on the fact that agile methodologies are more efficient as compared to traditional methodologies such as IBM RUP, Prince2 and CMMI (Deepti and Alok, 2011). This can better be explained through limitations of agile software development as described below.

There is difficulty in fitting agile with organizational culture

Application of the software requires much input from individual and team members. Agile development requires constant adjustments to the processes for the purposes of reflecting situations as hand. In this case, the software requires individuals to constantly switch roles as needed alongside adapting to work environment. Such a domain makes processes secondary to people (Eran and Hillel, 2011). Additionally, agile does not allow for application of narrow responsibilities, policies, processes and multi-purpose methodologies. Consequently, there is much difficulty in merging agile with some organization cultures since it requires identification of a team capable of working independently from the rest (Highsmith and Cockburn, 2001). Those included in such a team are always not subject to same rules as the rest of the organization. At the same time, the constituted team cannot fit well within traditional organizational culture. The implementation of the software requires change in organizational leadership and culture.

Limitations on the size of teams

Agile is considered a highly participative style of software development hence affecting efficiency of the processes owing to the number of participants involved. The software restricts the size of the team involved in a project which naturally extends project sizes. Barlow et al., (2011) asserts that agile team can be applicable in large projects where the project is sub-divided into relatively independent sub-projects. Such approach has negative consequences since it requires recruitment of higher-level project management capable of coordinating smaller teams (Rizwan, 2012). Development of agile software calls for sub-division of complex project based on architecture. Such division yields different results as opposed to division according to features.

Agile emphasizes on the use of collocated team

The software emphasizes on the use of face-to-face and spontaneous communication which limits its applications. In office environment it requires that the available space has to accommodate all stakeholders involved in daily activities. This is since each team member has to be within face-to-face reach of all members to facilitate communication. Such arrangement makes it difficult for companies to use agile since most organizations are organized by departments hence quite impossible to move people around based on agile projects. According to Pelrine (2011), the situation may seem impossible when it comes to coordination amongst developers who are often distributed throughout organization. Therefore, agile collocation principle does not comply with subcontracting within organizations.

There is limitation in process definition

Unlike other applications, agile methodology in most cases does not define

project management processes (Cockburn, 2000). For instance Scrum methodology does not give details on the processes involved. In contrast software development methodologies involve various processes which include analysis, architecture, implementation, management of project amongst other processes (Laanti et al., 2011). This makes those responsible for managing agile teams more of team leaders than project managers. Therefore, most agile methodologies are efficient in managing day-to-day team management activities.

Emphasis on team ownership rather than individual accountability

Agile development focuses on the importance of team ownership and team improvement for successful results. However, most organizations use performance-reward system for the purposes of assessing individual performance leading to individual rewards. The software overlooks individual performances within teams hence denying individuals opportunities on craving for high performances. Moreover, efficiency within organization demands that both perspectives on teamwork and individual input should be accounted for (Zhang and Shailesh, 2011).

Inappropriate Documentation

According to Eran and Hillel (2011), agile movement focuses on having documents with lower cost of production as compared to benefits. According to agile principles face-to-face communication replaces documentation. However, this creates a problem when dealing with multiple teams separated by distance. Such instance requires the necessity of capturing some knowledge in documents for other to access in case there are no opportunities for face-to-face communication. Documents are often

associated with various models, diagrams and manuals and other types of crucial information that can easily be forgotten. Lack of such models is a disadvantage to project managers and product owners since there is need for communication with developers. Therefore, agile provides risk on delivery of poor quality information that is inefficient in supporting future decisions.

Limitation on knowledge and training

Agile relies on tacit knowledge contributed by the team hence does not require uniformly high-capability people. This eliminates the idea on writing the knowledge down as documentation. Such process may lead to architectural mistakes not easily detected owing to lack of documentation. There is lack of investment in highly trained life-cycle architectures and plans capable of reducing the risk on documentation. The success of agile methods relies on a team of good people rather than skillful practices and principles. There is not much emphasis on user interface design and usability (Dybå and Dingsøyr, 2008).

References

Barlow, J. B., Keith, M., Wilson, D. W., Schuetzler, R. M., Lowry, P., Vance, A., & Giboney, J.

(2011). Overview and Guidance on Agile Development in Large Organizations. Communications of The Association For Information Systems, 2925-44

Cockburn, A. (2000). The agile software development, Highsmith Series.

Deepti, M., &Alok, M. (2011). Complex software project development: agile methods adoption,

Journal of Software Maintenance and Evolution: Research and Practice, 23(8), 549-564.

Dybå, T., & Dingsøyr, T. (2008). Empirical Studies of Agile Software Development: A

Systematic Review, Information and Software Technology (2008), doi: 10. 1016/j. infsof. 2008. 01. 006

Eran, R., & Hillel, R. (2011). Supporting agile software development through active

documentation, Requirements Engineering, 16(2): 117-132.

Highsmith, J., & Cockburn, A. (2001). Agile software development: the business of innovation,

34 (9): 120-127.

Laanti, M/. Salo, O., & Abrahamsson, P. (2011). Agile methods rapidly replacing traditional

methods at Nokia: A survey of opinions on agile transformation, Information and Software Technology, 53(3): 276-290.

Pelrine, J. (2011). On understanding software agility – a social complexity point of view, E. CO

13(1-2): 26-27.

Rizwan, J. Q. (2012). Agile software development methodology for medium and large projects,

Software, IET, 6(4): 358-363.

Zhang, Y., & Shailesh, P. (2011). Agile model-driven development in practice, Journal IEEE

software, 28(2): 84-91.