Overscoping in Agile Developments – Zachary Zerafa (24557656)

Abstract

With the growing demands for software products, the demand for strong programming frameworks also develop. Several software development philosophies have developed in the past half century, a notable example being agile development's scrum and kanban methods. These methodologies are designed to be highly adaptable to changes in requirements and constrains; often containing iterations of short-term activities to efficiently increment workload, however these method ar susceptible to the issue of overscoping, which may lead to unintended side effects that in certain contexts could potentially jeopardise the efficiency, quality, and cost of the development process. This article, the causes, effects and gravity of overscoping will be assessed and a range of possible mitigation tactics will be discussed.

1 – Introduction

What is Overscoping

Overscoping (also referred to as scope-creep) refers to attempting to fufil more stakeholder requirements than are financially, technically or chronically feasible by an assigned development team. This is often convoked by unrealistic expectations on schedule and budget or otherwise encompassing an "unrealistically grand amount of functions" (Bjarnason et al., 2012).

There are various sources from which overscoping can occur, one of which includes the misunderstanding of the process on the end of stakeholders, as unfamiliarity with the nature of their desired system may induce scope creep; the crescendo of requirements building up to a level that becomes unmanageable.

Overscoping may also arise due to the diverse visions of individual stakeholders not being able to come to a consensus on a defined set of requirements to implement in a program (Bjarnson et al. 2010), which can in turn lead to delays due to the lack of direction for the development team. An array of different agile methods are vulnerable to the effects of overscoping, either due to their communication-independence, common misimplementations or lack of definitive deadlines or project constraints.

Overscoping in Scrum

Scrum is an agile methodology that employs a cyclic process called a 'sprint', typiclly lasting two to four weeks (Ionel 2008), where developers develop and display a prototype of the software product, following a discussion with stakeholders on which requirements are fufilled or need additional work. This always leaves developers with a definitive goal that directly satisfies the requirements set by stakeholders.

Though this leaves developers with a strong sense of direction (which can aid in avoiding overscoping), due to the abiguity of a project's duration and high frequency of stakeholder meetings (Ionel 2008), requirements are being updated at a regular interval without a static date to compare with for time management.

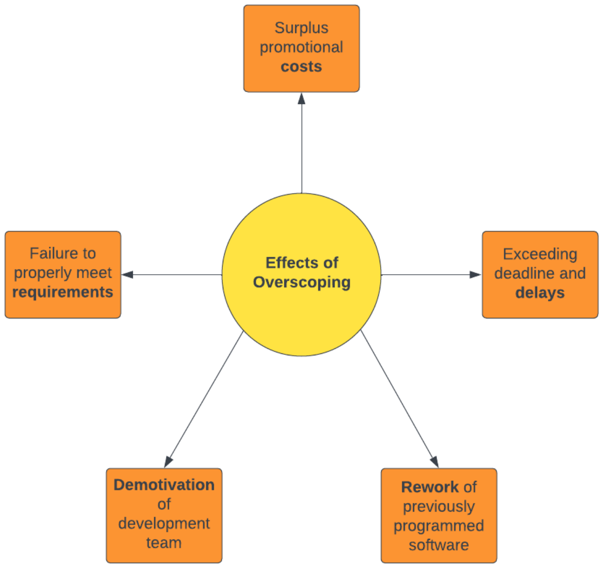
Overscoping in Kanban

Kanban is another agile method that uses a backlog nicknamed the “kanban”, which categorises of all stakeholder requirements according to their level of completion to efficiently allocate resources amongst different requirement tasks autonomously from stakeholders; it has abismal reliance on constant stakeholder communication. However this method is also tainted by overscoping. Kanban contains few stakeholder meetings aside from an initial instance of communication with stakeholders in order to form the backlog, which can prove problematic when a team practically recognises that the user stories may have a scope that is unrealistic to implement under certain conditions (Bjarnason et al., 2012). In such a situation overscoping arises from a lack of communication between developer and stakeholder.

2 – Quality of the Issue of Overscoping

Risks of Cost and Delays

There is ample reason for coverscoping to be researched and analysed; the risks of being unaware of this issue can lead to large losses in resources and time, which can only be prevented by strategising management strategies and assessing their practicality through case studies. Overscoping is known to impediment projects financially (Bjarnason et al., 2010) and delay the release of a several product, since the misallocation of resources that could go towards the final product are rendered futile and reworking on previous tasks often becomes necessary.



For instance, a project release date that isn’t properly correlated with the quantity and intensity of the requirements chosen by stakeholder may create a time window that is not feasible for development. This generic case of overscoping leads to extending the deadline due to a misunderstanding of the requirements as well as extra costs in promotional campaigns (Bjarnason et al., 2012).

An instance such as the previously explained has been viable material for research in the past, as a case study by Prasanta Dey (Dey et al. 2007) applied risk-mapping methods to quantify the gravity and probability of a range of known software
development risks and concluded that scope creep had the highest severity out of all software development risks, further stressing the cruciality of research into strategies to combat overscoping.

The effects of overscoping on financial resources, timing, the development team and the stakeholders as discussed in Overscoping: Reasons and consequences (Bjarnson et al. 2010). The numerous undesired effects render overscoping the prime danger in software development projects

High Frequency of Issue

Though it is clear that ignorance of overscoping poses a great threat to the software development industry, the saliency of this issue in a large quantity of case studies emphasises the need for this issue to be addressed.

A case study with an anonymous software companry with approximately 5000 employees was conducted by Lund University to investigate the root causes of overscoping, where a sample of 19 employees recounted five previous cases of overscoping in their history with the company (Bjarnson et al. 2010). This demonstrates that scope creep is a highly practical issue that is present in many project scenarios, offering incentive for research.

Another case study including a team of students being taught how to carry out various agile methods found the outcome that many teams experienced scope creep, despite efforts of maintaining strong stakeholder communication, and found an uneven distribution of workload as a correlated effect (Anslow et al. 2015).

Feasibility to Mitigate

Though there is a high presence of overscoping in the software industry, there are many methods to ward off scope creep that are efficient and cost-effective to implement, which being discussed in research can help raise awareness and boost overall productivity for all software development teams. Agile methods by themselves tend to have higher resistance to overscoping than traditional waterfall methodologies (Bjarnason et al., 2012), however the sole use of agile development methods does not suffice in eradicating the issue, as visible in the previous student case study (refer to *High Frequency of Issue*). A range of extensions for agile methods are plausible solutions for overscoping, such as cost estimations and enhancing methods of communication.

3 – Cost Estimate Solution

$$\sum_{i=1} c_i = \sum_{i=1} t_i k$$

The cost (c) of a requirement (i) is proportional to the time (t) spent on the requirement (Slittili et al. 2011); this is an example of quantifying cost estimation to approximate the resources of a requirement and whether it is feasible.

What is Cost Estimation

Cost estimation refers to techniques used by development teams to predict the magnitude of requirements that a stakeholder provides (Pfleeger et al. 2005). The methods used are often mathematical in nature, using previous statistics of previous case studies and experiences as data to run analysis tests such as neural networking or regression (Jorgensen et al. 2007). Cost estimation offers great defense to development teams against the risks of scope creep as it offers a way to measure the resources and project duration that requirements require and allows for comparison to what the team is capable of undertaking; allowing overscoping requirements to be identified early.

Requirements

Cost estimation is enhanced by a wide range of analysis tools and techniques used to make optimum use of the data at hand. UML (Unified Modelling Language) refers to a standard of representing software systems through object-oriented illustrations (Kim et al. 2006), and serves as a useful tool to project the cost of a project at a technical level. UML use case diagrams demonstrate a variety of different users and the actions that they would exexute through the software system (Pfleeger et al. 2005) (also known as use cases), and can be manipulated for cost estimation by finding the sum of the actors and use cases as a quantity to compare to previous projects and their correlated costs, assisting in deciding whether the project is within the capabilities of the development team.

Regression is a mathematical tool that is frequently used in software cost estimation that works on mapping bivariate data with a line of best fit (Pfleeger et al. 2005). This can be used to compare a set prior projects and their costs, graphed based on cost as well as anothere chosen factor of the projects. When a line of best fit is rendered (Jorgensen et al. 2007), the relationship between the factor of the project and cost can be examined, for instance, examining the rate that the cost of a project increases per requirement.

Limitations and Solution Space Strategy

Cost estimation provides an edge in hasty identification of overscoping, however it is highly dependent of data originating in previous projects and planning schematics such as UML, which may not always be conveniently available to software development teams. Additionally, it is possible that the data is unreliable as it may be affected by an uncountable array of external factors that may not be applicable to the project in context.

In applying this solution, it is worth noting that cost estimates are always prone to inaccuracies (Pfleeger et al. 2005). A healthy solution space strategy would be to account for size of the dataset, chosen confidence interval for statistical calculations and the context of the current project in contrast to those in the dataset as a general measure on how reliable the output of cost estinamtion techniques are.

4 – Solution of Communication

Relationship between Communication and Overscoping

Overscoping in select contexts arises as a product of lack of communication with stakeholders; either by misinterpreting or not recording key stakeholder requirements (Ajmal et al. 2019), and in turn leads to the development team undertaking more requirements than necessary. For this reason, it becomes imperative that regardless of the type of agile method utilised, a strong communication line with stakeholders is maintained.

Methods such as Kanban have minimal communication and as a result fall into this category of risk, however Scrumban is a Kanban-specific solution that attempts to integrate the iterative communication of scrum into the Kanban method as to rectify any scope misunderstandings immediately. It functions by maintaining the kanban backlog but setting intervals for stakeholders to analyse and question the state of the backlog, allowing for dialogue that will help hone the development team on only the scope defined by the stakeholders (Bhavsar et al. 2020).

Requirements

An assortment of tools and strategies can be implemented to establish a strong network with stakeholders, including JIRA; a development management service that offers a collection of features such as a digital kanban backlog, sprint progress reports, a custom query language to search for specific sprints, in conjunction with a abundance of graph generators that visualise all the aforementioned data (this can be used in conjunction with cost estimation techniques to have a high degree of control over the project’s scope). JIRA provides all the necessary means to construct and describe the progress in a scrumban framework, which provides a strong basis in communicating the project with stakeholders through JIRA’s repertoire of graphing presentation features.

Limitations

Ameliorating communication with stakeholders is a cost efficent and trivial solution to apply, however a few caveats are present. Being closer knit with stakeholders increases the risk of stakeholders adding more requirements as the project slowly progresses due to the facilitated communication. Stakeholders that are unaware of the issue and consequences of overscoping can inadvertently increase the workload for their project and face the consequences of decreased quality and extended deadlines.

With this in mind, the solution space would require setting clear boundaries with the stakeholders on what is feasible and the limit to the quantity of requirements. Informing stakeholders about the concept of scope creep creates a common understanding that protects both parties from the risks that overscoping suscitates.

Summary

Overscoping is a serious dilemma in the field of software development that can plague even robust agile methodologies with decreased project quality and deadline extensions, however there are a range of mitigation tactics such as cost estimation and extensive communication that prove effective in keeping scope creep at bay. It is often suitable to apply a balanced mix of these solutions rather than one solution alone in order to have a strong buffer against scope creep and secure a project's efficiency and quality.

Reference

Ajmal, M., Khan, M., & Al-Yafei, H. (2019). Exploring factors behind Project SCOPE CREEP
stakeholders' perspective. *International Journal of Managing Projects in Business*, *13*(3), 483–504. https://doi.org/10.1108/ijmpb-10-2018-0228

Anslow, C., & Maurer, F. (2015). An experience report at teaching a group based Agile Software Development Project Course. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. https://doi.org/10.1145/2676723.2677284

Bhavsar*, K., Shah, D. V., & Gopalan, D. S. (2020). Scrumban: An agile integration of scrum and Kanban in software engineering. *International Journal of Innovative Technology and Exploring Engineering*, *9*(4), 1626–1634. https://doi.org/10.35940/ijitee.d1566.029420

Bjarnason, E., Wnuk, K., & Regnell, B. (2010). Overscoping: Reasons and consequences - a case study on decision making in software product management. *2010 Fourth International Workshop on Software Product Management*. https://doi.org/10.1109/iwspm.2010.5623866

Bjarnason, E., Wnuk, K., & Regnell, B. (2012). Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering. *Information and Software Technology*, *54*(10), 1107–1124. https://doi.org/10.1016/j.infsof.2012.04.006

Dey, P. K., Kinch, J., & Ogunlana, S. O. (2007). Managing risk in software development projects: A case study. *Industrial Management & Data Systems*, *107*(2), 284–303. https://doi.org/10.1108/02635570710723859

Ionel, N. (2008). *Critical Analysis of the Scrum Project Management Methodology*.

Jorgensen, M., & Shepperd, M. (2007). A systematic review of Software Development Cost Estimation Studies. *IEEE Transactions on Software Engineering*, *33*(1), 33–53. https://doi.org/10.1109/tse.2007.256943

Kim, S. E., Lively, W., & Simmons, D. (2006). *An Effort Estimation by UML Points in the Early Stage of Software Development* . https://doi.org/10.12691/ajse-1-1-2

Pfleeger, S. L., Wu, F., & Lewis, R. (2005). *Software cost estimation and sizing methods: Issues, and guidelines*. Rand Corp.

Sillitti, A., Hazzan, O., Bache, E., & Albaladejo, X. (2011). *Agile processes in software engineering and Extreme Programming: 12th International Conference, Xp 2011, Madrid, Spain, May 10-13, 2011. proceedings*. Scholars Portal.