M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 181

Impact of Hypothesis-Driven Development on Effectiveness, Quality, and Efficiency in Innovation Projects

Sebastian Klepper[1] Bernd Bruegge[2]

Abstract: We evaluate the impact of hypothesis-driven development in innovation projects dealing with complex problems, focusing on decision making processes instead of experimentation methods. Our findings from multiple qualitative case studies show that this type of empirical research used for decision support can enhance effectiveness and quality without negatively impacting efficiency.

Keywords: Hypothesis-driven development; Decision support; Continuous innovation; Continuous experimentation; Project success factors

1 Introduction

The latest Annual State of Agile Report[3] indicates that there are multiple success factors for innovation projects which drive organizations to adopt agile methodologies in the hopes of maximizing their overall productivity: Organizations want the ability to react to change while delivering the right scope with great speed, maximize business value as well as product quality and user satisfaction. In short, they expect agile projects to strike the perfect balance between all possible constraints. Interestingly, while almost all respondents are able to realize improvements in efficiency, about a quarter of them fail to achieve reduced project risk, improved quality, or better business/IT alignment. A consequential trend seems to be to revert to planning to better control risk and results. Popular activities include story mapping, iteration and release planning. Even road-mapping and portfolio planning have entered the picture, possibly related to increasingly large organizations adopting agile methodologies.[4]

This indicates a desire to gain more control of project performance and outcome and begs the question if there are better ways to achieve this than re-introducing planning activities. Our hypothesis is that instead of through planning we can improve project performance or success through better decision making processes and an empirical approach to problem solving. Related case studies in this area have studies the introduction of continuous experimentation

[1] Technische Universität München, Munich, Germany sebastian.klepper@tum.de
[2] Technische Universität München, Munich, Germany bruegge@in.tum.de
[3] VersionOne Inc.: 11th Annual State of Agile Report, 2017, URL: https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2
[4] also see VersionOne Inc.: 10th Annual State of Agile Report, 2016, URL: https://explore.versionone.com/state-of-agile/versionone-10th-annual-state-of-agile-report-2

in large organizations and discovered that, besides the challenge of experimenting in production, there is a fundamental conflict with the established way of working and thinking. [RM15; Ya17] "Company philosophy or culture at odds with core agile values" is also reported as the primary challenge when adopting and scaling agile processes.[3] We therefore focus on relevant decision making processes which form the basis for agile exploration as well as reactivity in the development processes. [see FS17; Ri11]

To judge impact on project success we must first define how to measure it. There have been many attempts at describing project constraints and success factors. The most prominent example is the "iron triangle" of scope, time, and cost with quality as an emergent property of the result. Over time, other factors have been described such as system quality, service quality, user satisfaction, learning effects, impact on stakeholders, organizational benefits. [At99; DM03] Bloomberg [Bl13] proposes an extension to the historic iron triangle in order to transform it to agility, quality, time. Ralph; Kelly [RK14] try to summarize the plethora of individual factors in a "software engineering success framework": Projects produce artifacts which perform in markets; projects exhibit efficiency, artifacts exhibit quality, and markets exhibit performance. Following these updated models, we differentiate *effectiveness*, *quality*, and *efficiency* as dimensions of project success.

2 Hypothesis-driven development

Our methodology of choice to improve decision making processes is hypothesis-driven development (HDD) – essentially the application of the scientific method to software product development. This school of thought found its way into software engineering by way of the "lean startup" movement popularized by Ries [Ri11] and further refined in a collaborative note with Eisenmann et al. [ERD11] that provides guidelines to formulating, evaluating and acting on hypotheses. A *hypothesis* in this context is any assumption critical to project success that is falsifiable, i.e., it can be validated by means of an experimental method.

This approach has gained traction in the world of empirical software engineering and multiple models have been proposed, either high-level process models or methods for specific types of experimentation. They focus on either initial exploration or continuous development and use hypotheses correspondingly to capture assumptions about business strategy, feature value, or feature refinement. Necessary data is collected using techniques like customer feedback, prototypes, or analytics data from live experiments. [Bo13; Bo14; EÞ16; Fa17; OB15]

It is important to differentiate between the mechanics of experimentation and the higher-level decision making process. At the core of our investigation is decision support, meaning different types of decisions such as triage, prioritization, selection, or scheduling. [AW05; HKP15, see] These decision problems are continuously identified and supporting empirical evidence is produced to address uncertainty and corresponding risk.

3 Case studies

To investigate impact of hypothesis-driven development on effectiveness, quality, and efficiency, projects applied the methodology over the course of several weeks. In total, we analyzed 10 projects with a total of 69 developers for a total of 120 weeks, amounting to 3,920 person days of evaluating hypothesis-driven development. Fig. 1 and 2 show the distribution of developer count and evaluation period across projects.

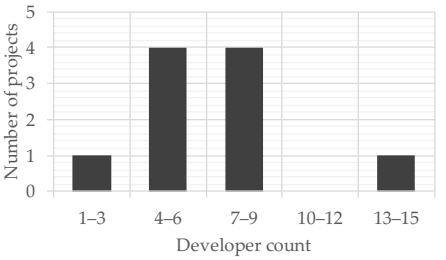

Fig. 1: Distribution of developer count

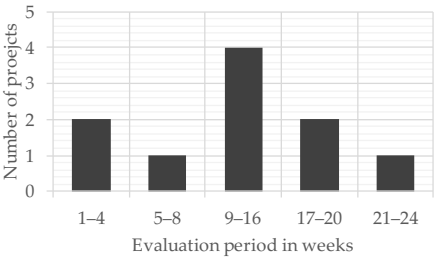

Fig. 2: Distribution of evaluation period

Subsequently projects qualitatively evaluated success criteria using personal opinion surveys in combination with data from issue trackers. [see also SSS08, pp. 35-62, 63-92] We also selected suitable projects with similar characteristics within each organizations to serve as a control group. These control projects also used agile methodologies but did not apply HDD and reportedly suffered from the aforementioned lack of quality in decision making.

To help projects apply hypothesis-driven development quickly and consistently we provided them with relevant literature as well as a custom issue type in JIRA[5]. This also allowed collection of structured data about experiments in correlation with regular development tasks.

4 Findings

Collecting and consolidating data and feedback from all cases studies allowed us to study differenct aspects of how hypothesis-driven development impacts project success. We subsequently present our findings grouped by the success dimensions defined initially.

4.1 Impact on effectiveness

Our first and foremost research question is: **Does hypothesis-driven development allow teams to solve problems more effectively?** We reason that frequently validating critical

[5] `https://www.atlassian.com/software/jira`

assumptions enables teams to make *better* decisions *earlier* in time. This should reduce risk posed by uncertainty since misconceptions are recognized faster and less time is wasted following a wrong path, presumably leading to better problem/solution fit as well as faster discovery of the right solutions.

A qualitative survey amongst project stakeholders captured feedback about solution fit, covering two important aspects:

1. How good does the solution match the problem?
2. How good is the solution compared to other available solutions?

In addition to "absolute" solution fit we are also measuring the aspect of innovation (finding new and better solutions). Fig. 3 and 4 show these ratings on a scale of 1 (excellent) to 6 (deficient). The majority of stakeholders rated both solution fit and innovativeness 1 (excellent) or 2 (good) with a respective mean of 1.85 and 1.75, indicating that projects indeed achieved a good problem/solution fit.

However, there is a noticeable second peak at 4 (adequate) in both dimensions. Further investigation revealed that these ratings stem from advanced users with undiscovered special needs. Their below-average rating conveys that the feature set is not "satisfactory" (3) yet.

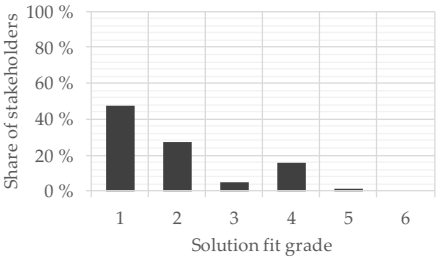

Fig. 3: Distribution of rating of solution fit

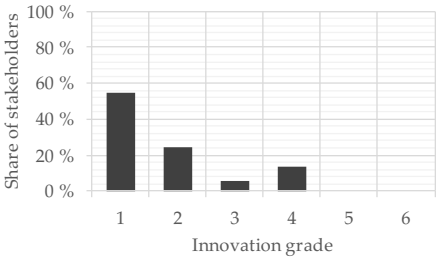

Fig. 4: Distribution of rating of innovativeness

Anecdotal evidence additionally suggests that testing of critical assumptions led yielded the following benefits to project members:

- Structured overview of facts and areas of uncertainty,
- conscious evaluation of decision components such as priority, effort, or alternative,
- easier decisions once comfortable level of confidence was reached,
- faster discovery of mistakes and less changes or pivots,
- implicit generation and documentation of rationale of these decisions.

On the other hand, critical respondents mentioned that empirical evidence is often trumped by other factors such as urgency, taste, or politics, raising the question whether additional research effort is justified. Sect. 4.3 addresses this concern about efficiency.

4.2 Impact on quality

In addition to higher effectiveness, we hypothesize that better decision making results in a solution of high quality. Timely recognition of wrong assumptions should result in fewer pivots, changes, or reverts and therefore more time being spent on proper execution and refinement. Our research question is: **Does hypothesis-driven development enable teams to also produce a high-quality solution?**

Using a qualitative survey, we collected feedback from different stakeholders of the projects, ranging from users to managers to technical experts. We asked for a rating of specific qualities such as ease of use, intuitiveness, performance, and robustness on a scale of 1 (excellent) to 6 (deficient). Due to the wide range of stakeholder type and quality factors we grouped feedback into two categories: Fig. 6 shows rating of technical quality based on performance, stability, and maintainability; Fig. 5 shows rating of usability based on visualization, intuitiveness, and ease of use.

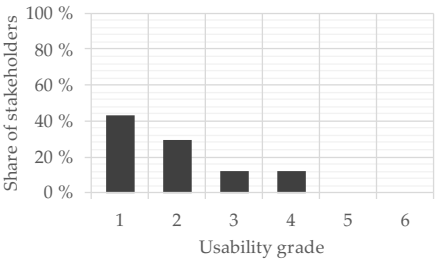

Fig. 5: Distribution of rating of user satisfaction

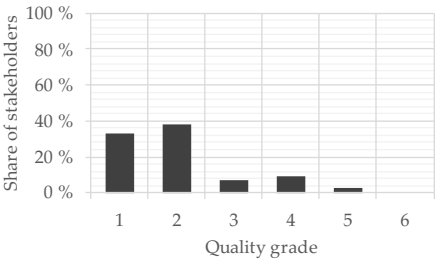

Fig. 6: Distribution of rating of technical quality

With the distribution skewed towards 1 (excellent) and 2 (good), projects achieved a mean rating of 1.81 on technical quality and 1.85 on usability. We consider this an above-average result, which is also backed by stakeholder feedback and the results of the Annual State of Agile Report where 25 % of respondents reported no improvement regarding software quality and 36 % regarding maintainability.[3]

This notably is an absolute and qualitative measurement of subjective criteria and not the quantitative result of a controlled experiment. However, anecdotal evidence from project stakeholders suggests that, compared to other projects in their respective organization, this approach indeed results in an improved use of time and therefore attention to non-functional requirements such as usability and sustainability.

4.3 Impact on efficiency

A serious concern was that HDD could negatively impact efficiency since it involves additional effort for designing, conducting, and evaluating experiments. As a counter-

argument, this additional effort could also be outweighed by time-savings through of increased effectiveness.

To investigate this, projects reported both cycle time and lead time before, during, and after the evaluation period. We differentiate between work items regarding functionality (e.g., user stories) and experiments themselves. To control for external influences or fluctuations load, we collected the same data from control projects. The collected data give an indication that efficiency is not negatively affected, in particular regarding the following questions:

Does HDD slow down development of functionality? Comparison of evaluation periods with non-evaluation periods both within the same projects and with control projects showed no significant shift in cycle time of other work items.

Does HDD slow down start of development of functionality? Similar analysis of lead times showed no significant differences in how much time work items were spending in the backlog before being picked up for development.

Does cycle time of experiments differ from other work items? Average cycle time for experiments was the same as for other work items. Interestingly, similar to outliers in development time some experiments took longer than expected to collect the required amount of data.

Fig. 7 shows a particularly demonstrative example of a project with stable cycle time over 28 weeks, unaffected by an 18-week evaluation period of hypothesis-driven development.

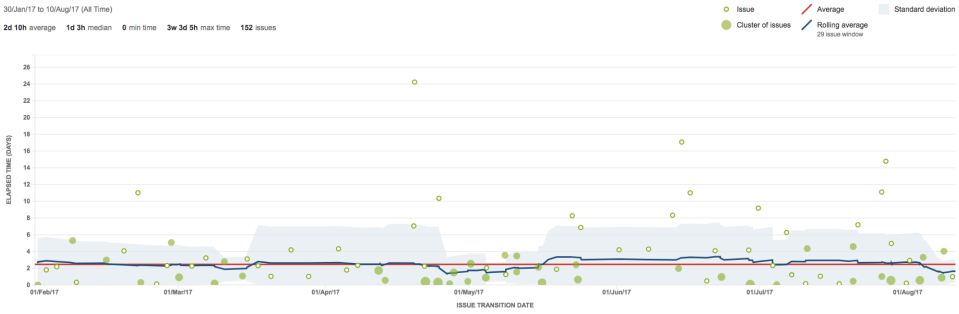

Fig. 7: JIRA control chart showing stable cycle time with and without HDD

5 Limitations

We presented the result of multiple independent experiments that were primarily qualitative in nature and not strictly controlled. However, initially having to dissect the aspect of "performance" or "success" into separate dimensions is an indicator for interplay of multiple influencing factors. In industry projects these are hard to isolate but we made an effort to

conduct evaluation during undisturbed periods and selected additional projects as a control group.

In our surveys and data analysis we tried to control for cross-influences. The line between effectiveness and quality is particularly blurry since both poorly executed features and the wrong feature set presumably both cause low satisfaction. Stakeholder surveys also are subjective and might suffer from selection bias as well as confirmation bias. Additionally, quality metrics are missing mantainability concerns like code-quality and test coverage.

Evaluation time per project was limited, ranging from two weeks to six months. This might skew results due to project members either not being familiar enough with the techniques yet or certain types of problems not having occurred yet. Longer evaluation periods and multiple evaluation points might alleviate this effect.

6 Conclusion

We studied how the emerging trend of empirical research in software engineering impacts project success. As both a contrast and supplement to related case studies, we focused on hypothesis-driven decision support instead of methods for experimentation. Our findings indicate that hypothesis-driven development enhances the quality of decisions and helps projects deal with uncertainty while not losing the ability to react to change. In particular, problem solving is enhanced by allowing to recognize misconceptions earlier and discover the right solutions faster. This leads to better problem/solution fit, overall quality of software, and usability since less time is spent on correcting mistakes. The same effect also appears to reduce negative impact on efficiency – the primary concern we faced when introducing the methodology. For more reliable and generalizable insights into the benefits and drawbacks of hypothesis-driven development, we pursue long-term quantitative evaluation.

References

[At99] Atkinson, R.: Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. International Journal of Project Management 17/6, pp. 337–342, 1999.

[AW05] Aurum, A.; Wohlin, C.: Engineering and Managing Software Requirements. Springer, 2005.

[Bl13] Bloomberg, J.: The Agile Architecture Revolution: How Cloud Computing, Rest-Based SOA, and Mobile Computing are Changing Enterprise IT. Wiley, 2013.

[Bo13] Bosch, J.; Olsson, H. H.; Björk, J.; Ljungblad, J.: The early stage software startup development model: A framework for operationalizing lean principles in software startups. In: Lecture Notes in Business Information Processing. Vol. 167, pp. 1–15, 2013.

[Bo14] Bosch, J.: Continuous Software Engineering. Springer, 2014.

[DM03] Delone, W. H.; McLean, E. R.: The DeLone and McLean Model of Information Systems Success: A Ten-Year Update. Journal of Management Information Systems 19/4, pp. 9–30, 2003.

[ERD11] Eisenmann, T.; Ries, E.; Dillard, S.: Hypothesis-Driven Entrepreneurship: The Lean Startup. Harvard Business School Background Note 812-095 44/December, pp. 1–23, 2011.

[EÞ16] Ekström, M.; Þorvaldsson, Í. D.: Predicting and Analyzing Feature Value when R&D is an Experiment System, PhD thesis, University of Gothenburg, 2016.

[Fa17] Fagerholm, F.; Sanchez Guinea, A.; Mäenpää, H.; Münch, J.: The RIGHT model for Continuous Experimentation. Journal of Systems and Software 123/, pp. 292–305, 2017.

[FS17] Fitzgerald, B.; Stol, K. J.: Continuous software engineering: A roadmap and agenda. Journal of Systems and Software 123/, pp. 176–189, 2017.

[HKP15] Hesse, T.-M.; Kücherer, C.; Paech, B.: Experiences with Supporting the Distributed Responsibility for Requirements through Decision Documentation. Softwaretechnik-Trends 35/1, 2015.

[OB15] Olsson, H. H.; Bosch, J.: Towards continuous customer validation: A conceptual model for combining qualitative customer feedback with quantitative customer observation. In: Lecture Notes in Business Information Processing. Vol. 210, pp. 154–166, 2015.

[Ri11] Ries, E.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business, 2011.

[RK14] Ralph, P.; Kelly, P.: The dimensions of software engineering success. In: Proceedings of the 36th International Conference on Software Engineering - ICSE 2014. Pp. 24–35, 2014.

[RM15] Rissanen, O.; Münch, J.: Continuous Experimentation in the B2B Domain: A Case Study. In: Proceedings - 2nd International Workshop on Rapid Continuous Software Engineering, RCoSE 2015. Pp. 12–18, 2015.

[SSS08] Shull, F.; Singer, J.; Sjøberg, D. I.: Guide to advanced empirical software engineering. Springer London, 2008.

[Ya17] Yaman, S. G.; Munezero, M.; Münch, J.; Fagerholm, F.; Syd, O.; Aaltola, M.; Palmu, C.; Männistö, T.: Introducing continuous experimentation in large software-intensive product and service organisations. Journal of Systems and Software/, 2017.