

An Empirical Study of the Software Development Process, Including Its Requirements Engineering, at Very Large Organization: How to Use Data Mining in Such a Study

Colin M. Werner^(✉) and Daniel M. Berry

Cheriton School of Computer Science, University of Waterloo,
Waterloo, ON N2L 3G1, Canada
colinwerner@gmail.com, dberry@uwaterloo.ca

Abstract. Very Large Organization (VLO) develops and manufactures hardware and software products, with each product being developed in its own project. Each project, from its inception, maintains a database that contains a wealth of data pertaining to its software development lifecycle. To empirically study VLO's software development process, the authors mined the data from seven consecutive VLO projects to determine whether the data exhibit any anomalies and whether these anomalies can help assess a project's level of success. Some anomalies provide evidence of what VLO does well, while other anomalies highlight possible areas of improvement. Through the anomalies in the mined data, the organization can direct additional focus and research to specific areas of the development process, particularly its requirements engineering, to improve the likelihood of success for future projects.

While describing the results of the empirical study, the paper also shows how such a study can be conducted even when the mined data are not very detailed.

Keywords: Code repositories · Industry · Project management
Prototyping · Requirements engineering · Software engineering
Empirical case study

1 Introduction

Very Large Organization (VLO) is a mature organization that has released several products. Some of the products were deemed by the market to be successful, while others were deemed less successful. The authors were interested in determining the causes of success and of a lack of success. Fortunately, VLO, being a serious engineering firm, keeps a lot of data about each product, including, and in particular, its software development project. For each project, it keeps data from the project's inception, its requirements engineering (RE), its design, its implementation, its acceptance, through to its deployment, and beyond. These data have the potential to be mined to perhaps reveal key indicators of success and of its lack, a practice that is becoming more and more mainstream [1].

The purpose of the research reported in this paper was to explore whether these data could be mined in order to determine causes of success and causes of lack of success. The research started with mining the data of whole projects in order to not make any assumptions as to which steps of the process were critical, i.e., to let the data take us where they do. The mined data were very telling. Included in this paper are some of the key data points and conclusions drawn from the mined data, as well as advice on how to conduct such a data mining study when the data are not very detailed.

This paper is derived from the master's thesis of the first author [2]. Details, in particular, the actual data and its full analysis, that were left out of this paper in order to meet its page limit can be found in the thesis.

In the rest of this paper, Sect. 2 describes the typical product lifeline; Sect. 3 defines the mined data; Sect. 4 analyzes the data and draws conclusions; Sect. 5 offers some possible solutions to the observed problems; Sect. 6 summarizes the answers to the research questions raised in Sect. 4; Sect. 7 discusses threats and limitations of the conclusions; and Sect. 8 concludes the paper.

2 Product Description

VLO produces hardware-and-software products, each of which includes hardware, an embedded system, and a set of applications. Each product released by VLO has an associated development project, whereby multiple products may be developed in the same project. Generally speaking, when VLO releases a new product, the product consists of a new hardware design-and-construction accompanied by its own new embedded system and an applications layer. The software release model used for the majority of releases is based on an iterative waterfall model, in which each subsequent project is built directly on top of its previous project.

Many consumers worldwide use VLO products. However, these consumers do not buy directly from VLO. Instead, VLO sells its products to customers, and these customers resell the products to the consumer. Each development project to develop at least one product has a set of customers, who are the ones that determine whether a delivered product is accepted. Each project's database logs a variety of data about each acceptance: its date, the name of its accepting customer, and a variety of other details pertaining to the various hardware platforms supported and software versions produced.

The initial phase of a project is to create the vision of the project's product. The vision roughly defines the hardware and software of the product along with some key factors differentiating it from those of other products. Once the product vision has been completed, the requirements phase begins.

The requirements phase at VLO starts with the product marketing team, who are deemed to have substantial market knowledge regarding the product and interface directly with the customers. The product marketing team describes various aspects of the product in several market requirement documents, which form the basis for a market-viewpoint specification of the product's requirements. The project's development team reviews this requirements specification and negotiates with the product marketing team to finalize a set of requirements that will be implemented. This set of requirements is encapsulated in a project requirements document.

Once the product's requirements are believed to be understood, the development phase begins. During the development phase, if a substantial or unrealized change is required to the product, a change request (CR) must be created and sent through the necessary approval process. The intention of the CR process is to minimize requirements creep; however, a substantial number of CRs for a product could indicate problems in the RE process and could merit further investigation.

As a project is nearing completion, a near final milestone is for its customer to provide acceptance of the product. The acceptance does not mean that the development of the product is complete, as the acceptance may include conditions for small tweaks to the product. However, these conditions are usually small enough to consider the acceptance as the final clearance from the customer that the product's requirements have been met.

3 Data Compilation

Without industrial data, research like that described herein would lack external validity. Therefore, we were fortunate that VLO granted to the authors access to a number of tools in order to mine the data pertaining to each project. All of the data mining occurred after the release of all the products. Thus the mining was observational and nonintrusive. Due to the confidential nature of the products and projects, and for the protection of VLO, each datum was anonymized. At the time the data were mined, each datum was associated with one of seven consecutive projects. Each dataset consisted of a series of data from source-control repositories, CR reports, customer acceptances, employee opinions, and any other pertinent project information that was uncovered during the data mining.

Data were gathered until the authors felt that they could not learn anything new from additional data, i.e., until *saturation* was achieved. Clearly, recognition of saturation is highly subjective, but there is no other way to know when to stop mining data.

3.1 Project Lifelines

In order to form the dataset for a project, a project lifeline was created. The lifeline for a project simply records the duration of the project's development lifecycle, from beginning to end. The lifeline formed the basis for the rest of dataset, as the project lifeline created a natural axis on which to map the remaining data. A project's lifeline did not have a fixed length. As more data were discovered, the lifeline could be extended in order to include the additional data. Thus, the project lifeline became an implicit portion of the dataset.

3.2 Source-Control Repository

The source-control repository provided the largest set of data. Each project has its own branch in the source-control repository, allowing simple scripts to collect the data. A source-control commit is a single source commit by a single developer to the source-control repository. These data are simply counted as each source-control commit

occurred, so there is no advance notice of outstanding changes to be committed. A source-control commit may consist of a list of requested items, each of which is a feature request, a CR, a software bug fix, or a non-functional change such as an optimization or a change to a document. Therefore, the actions a developer may take for a single source-control commit can be any combination of (1) editing an existing file, (2) deleting an existing file, and (3) adding a new file to the source-control repository.

This paper does not classify the source-control commits based on content; for simplicity, each and every source-control commit is considered to be of equal value. This simplification may not be a fair evaluation and certainly deserves further attention [3]; however, the definition fits the purpose of this paper. As is typical with most source-control repositories, the data consisted of a series of source-control commits in each particular project branch. However, to ensure VLO's privacy, only the date of submission was retained during the data mining. The date of submission was sufficient information in order to keep track of the number of source-control commits in any project's lifeline.

3.3 CRs

At VLO, a process exists that must be followed whenever a new requirement or change in an existing requirement is desired, usually due to a customer's request. When a change is required, a CR must be submitted. A CR includes details of what is desired, why the change is required, and by when the change is expected. Normally, submission of a CR blocks its product's release. Once a CR has been submitted, the CR is first evaluated by a team of marketing experts to determine if the CR is worth pursuing. Next, the teams that would be involved in implementing the change must provide an impact analysis of the change. The analysis includes estimates for development, testing, localization, certification, or any other areas that may be affected by the change. Normally, it is very difficult to compile a complete list of all the teams that may be affected by a particular change. Therefore, any particular CR may include an incomplete list of estimates. This deficiency is itself a problem, as other work that was not in the original impact analysis is often discovered and consequently delays the completion of a CR. After the implementation information is gathered, the CR is forwarded to the CR committee, which ultimately decides whether to approve or reject the CR. The CR data collected for this paper describes only the approved CRs that were implemented for each product and not the rejected CRs. Moreover, the data about a CR do not include any details about the size, nature, or complexity of the change requested in the CR. So all we know about any CR is that it was approved and that its size, nature, and complexity must have been such that the CR was approved.

3.4 Customer Acceptances

VLO maintains a customer database which contains a variety of information related to each customer, including the date each product was accepted by the customer. For the purpose of this paper, this date shall be considered the date the product was fully accepted by the customer, even if the acceptance was conditional for any number of

reasons. The acceptance dates for each product were mined from this database and added to the dataset for the product's project.

3.5 Hardware

Every project did feature some new hardware. As part of the dataset, the number of hardware platforms per project was recorded. However, each hardware platform could also be classified into two different categories: revolutionary or evolutionary. Evolutionary hardware is typically providing an update to the hardware from a previous product. On the other hand, revolutionary hardware is ground breaking and carries with it a more substantial amount of novel technical innovation than does evolutionary hardware.

On occasion, the software developed for a new project could also be released on hardware from previous projects, also known as in-market hardware. Whether a project release included in-market hardware was recorded and added to the dataset.

3.6 Data Summary

The resulting project dataset includes from each of the seven projects

1. lifeline,
2. number of source-control commits,
3. number of approved CRs,
4. number of customer acceptances,
5. number of hardware platforms,
6. whether the project targets in-market hardware, and
7. whether the project includes revolutionary hardware.

4 Data Analysis

Some of the research questions (RQs) intended to be addressed by the work described in this paper are:

- RQ1:** Do the VLO project data say anything about the success or failure for any product?
- RQ2:** Do the VLO project data say anything about the *reasons* for success or failure?
- RQ3:** Are there any VLO project data that can serve as predictors of a product project's outcome [4, 5]?

The thesis [2] poses different, project-specific research questions, which are not answered here, because to do so would require more pages than are available. The interested reader is encouraged to download and read the thesis.

To answer these RQs, each project is compared with every other project. In addition, each project is compared with an ideal, but non-existent golden¹ project. The

¹ In empirical work, the *golden data* are the data to which each subject's data are compared to evaluate the subject's data. The researchers try to make the golden data completely correct, but know that they may not be.

golden project does not have any real data associated with it. Instead, the golden project is described by made-up data that represent the way we hope that every project would proceed, knowing full well that no project ever proceeds that way. The purpose of the golden project is to provide criteria that can easily be compared with any real project to provide answers to our research questions. The golden project

1. has a perfect, i.e., complete and consistent, requirements specification, but if not, the CRs are minor,
2. has all of its source code committed prior to any customer acceptances,
3. has all of its source code committed in the first half of the project,
4. has no CRs, but if a CR is required it should be resolved in the first half of the project,
5. has only one hardware platform,
6. has no in-market hardware platform, if the project involves a revolutionary hardware platform, and
7. has at most one revolutionary hardware platform.

Yes, these criteria *do* overlap.

The data about a project that are needed to evaluate how close the project is to the golden project are measures that allow the project to be compared to the golden project. These project data are, respectively,

1. the percentage of all code commits for the project that are done prior to the project's first customer acceptance.
2. the percentage of all code commits for the project that are done during the first half of the project,
3. the percentage of all CRs for the project that are resolved during the first half of the project,
4. the number of hardware platforms the project targets,
5. whether the project targets at least one revolutionary hardware platform,
6. whether the project targets at least one in-market hardware platform,

Note that the fifth criterion is decided by the fifth *and* sixth datum combined. Table 1 shows the values of the required data for each of the projects.

Table 1. Key data summary

	%-age code committed at first acceptance	%-age code committed at half way	%-age CRs resolved at half way	Number of hardware platforms	In-market hardware	Revolutionary hardware
Project 1	72.45	97.8	94.12	7	Yes	No
Project 2	88.15	95.4	74.19	4	No	No
Project 3	87.92	96.08	81.91	12	Yes	No
Project 4	72.05	94.53	90.23	2	No	No
Project 5	65.79	76.91	67.62	4	No	No
Project 6	67.88	78.88	71.62	2	No	Yes
Project 7	87.47	82.54	93.88	1	No	No

Comparing each project's data to the golden project's criteria reveals that no single project is quite a golden project, which was expected. For any project, the reason that the project fails to be a golden project depends on the project's circumstances and on the specific project data that deviate from the golden project criteria. In fact, a project with a single deviation² is likely to have a completely normal project lifeline. Charette points out that a failing project rarely fails for just one or two reasons [6]. Each of Projects 2, 4, and 7 has a single large deviation. However, none of them represents an extremity. In fact, each of these projects was typical for VLO and proves that VLO can successfully create incremental products. Ruhe *et al.* claim that a company's ability to create incremental products is a primary factor in the company's ability to produce large successful software products [7].

Each of the four remaining projects has multiple deviations and is far more interesting. Each of Projects 1 and 3, is one of the only two projects to target in-market hardware, thus explaining why these two projects had the most hardware platforms. The large number of hardware platforms explains each project's other deviations. In a typical VLO project targeting a new hardware platform, the new hardware platform would be released first. Only later would the versions running on in-market hardware platforms be released, thus explaining why the project's customer acceptances are scattered throughout the project's lifeline. The additional hardware platforms cause the project lifeline to be extended to cover the later in-market hardware platform releases; whereas a project without an in-market hardware platform release would be completed much sooner.

The two remaining projects, Projects 5 and 6 had by far the lowest data for the first three criteria. Closer examination shows that Project 5 had an additional hardware platform added approximately half way through its lifeline. As with Projects 1 and 3, the additional hardware platform extended the project's lifeline and caused other important data points to deviate even more. The added hardware platform can explain the deviations, but only if the originally planned hardware platforms were completed in a typical matter. Separating the statistics based on hardware platforms could be a worthwhile exercise but is outside the scope of this paper.

Project 6 is the most interesting, as it is the only project that was released on a revolutionary hardware platform. While the release on a revolutionary hardware platform can easily explain the deviations, it is more useful to use the data as indicators of *how well* VLO performed in the project. These indicators can be later used to help VLO perform better whenever it is implementing a revolutionary hardware platform and to bring VLO to the point that it is producing revolutionary hardware platforms on a routine basis.

In particular, when an organization is creating a product with a revolutionarily new hardware platform, the organization must perform adequate research, i.e., RE, prior to committing the revolutionarily new hardware platform to the product. The amount of research that VLO undertook before making the decision to include the revolutionary hardware platform in Project 6 is unknown. However, there is evidence that indicates that the prior research was not enough. There exists no strict criterion allowing a conclusion that enough research was performed. However, the poor outcome of Project 6 indicates that the technology was not thoroughly researched. That being said, there has been

² In the rest of this paper "deviation" means "deviation from the golden project criteria".

research on prototyping methods that can help to adequately and correctly identify requirements [8–12]. Typically, the point of prototyping is to allow starting early. DeMarco [13] indicates that all late projects have one thing in common: each started late³.

If a project is late, there is a good chance that the original effort estimates were inaccurate [14–16], which in turn, indicates that its software requirements were not well understood [17].

That doing thorough RE proved to be a factor in the success of a project was not surprising. There are other studies that have observed the same [e.g., 17, 18].

5 Possible Ways to Avoid Deviations

Whenever a project's requirements call for a revolutionary hardware platform, in addition to project's doing adequate research and prototyping, the project should minimize other new aspects of the product requirements. Minimizing other new aspects allows the project to focus on the revolutionary hardware platform and gives the project the best chance of success. Remember, it is very likely that a revolutionary hardware platform will create a number of dependent requirements. While these dependencies cannot be avoided, they create the necessity, and therefore, an additional motivation, to perform effective and efficient requirements analysis and management. According to Blyth *et al.* [19], the most crucial aspect of software engineering is the gathering of requirements. Much akin to prototyping, there are multiple methods that can be used in order to increase the effectiveness of the requirements specification, such as those proposed by Liou *et al.* and by Bowles [20, 21]. Finally, if a requirement is considered revolutionary, then perhaps VLO should consider following a slightly different development model, as Molokken-Ostvald *et al.* suggest [22].

The first author's thesis has a whole chapter devoted to a discussion of possible solutions [2].

6 Answering the Research Questions

The research questions driving the data mining study reported in this paper are:

- RQ1:** Do the VLO project data say anything about the success or failure for any product?
- RQ2:** Do the VLO project data say anything about the *reasons* for success or failure?
- RQ3:** Are there any VLO project data that can serve as predictors of a product project's outcome?

³ Lest the reader think that DeMarco's point is only a tautology, please note that DeMarco's observation is that in the typical late project he has known, the project was started well after the date that it should have been started in order to be able to guarantee finishing by the required deadline, everyone on the project knew that the project was starting late, and all on the project agreed to start the project anyway despite the certainty that the project had no chance to finish on time.

During the data mining, we were able to determine that

- RQ1:** the VLO project data do allow us to judge, by the absence or presence of project-and-VLO-specific secondary effects, the success or failure of any product,
- RQ2:** the VLO project data do allow deducing some of the reasons for any success or failure, and
- RQ3:** the most important VLO project data that can serve as predictors of a product project's outcome are the number and severity of the deviations from the golden project

7 Threats and Limitations

The biggest threats to the validity of the conclusions drawn from the data are the various assumptions and simplifications that were made to be able to conduct the analysis with the data that could be gathered. At each place such an assumption or simplification was made, it was described. On the other hand, since the project data were mined well after the fact, there is no chance (1) that these data could have been biased by a desire to support any particular hypothesis and (2) that the mining and the analysis could have affected the outcomes of the projects. Finally, as with any other case study, the results are about the particular case and cannot be generalized to other situations. However, multiple case studies on similar contexts coming to similar conclusions begins to indicate possible truths. So, we encourage others to do similar data mining studies on other software development projects.

8 Conclusion

Very Large Organization (VLO) clearly has the ability to release successful products. However, not all of the products are deemed by the market to be successful. The authors attempted to discover whether any predictors of success could be revealed through mining the available data from some of VLO's development projects. The data revealed a number interesting points, which can be used in predicting the likelihood of success for future products. In particular, there are clear indications that adequate prior research, prototyping, and RE are necessary to developing a successful product. There is a lot of literature, cited in this paper, that agrees with this finding. The authors have shown also that data mining can be used to reveal the causes of success and of a lack of success of projects in practical environments.

Acknowledgment. Daniel Berry's work was supported in part by a Canadian NSERC grant NSERC-RGPIN227055-15.

References

1. Hassan, A.E.: The road ahead for mining software repositories. In: *Frontiers of Software Maintenance (FOSM)*, pp. 48–57 (2008)
2. Werner, C.M.: A case study of a very large organization. Master's thesis, University of Waterloo, Waterloo (2012)
3. Agrawal, K., Aschauer, M., Thonhofer, T., Bala, S., Rogge-Solti, A., Tomsich, N.: Resource classification from version control system logs. In: *IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW)*, pp. 1–10 (2016)
4. Graves, T.L., Karr, A.F., Marron, J.S., Siy, H.: Predicting fault incidence using software change history. *IEEE Trans. Software Eng.* **26**(7), 653–661 (2000)
5. Engler, D., Chen, D.Y., Hallem, S., Chou, A., Chelf, B.: Bugs as deviant behavior: a general approach to inferring errors in systems code. In: *18th ACM Symposium on Operating Systems Principles (SOSP)*, pp. 57–72 (2001)
6. Charette, R.N.: Why software fails [software failure]. *IEEE Spectr.* **42**(9), 42–49 (2005)
7. Ruhe, G., Greer, D.: Quantitative studies in software release planning under risk and resource constraints. In: *International Symposium on Empirical Software Engineering (IS-ESE)*, pp. 262–270 (2003)
8. Hooper, J.W., Hsia, P.: Scenario-based prototyping for requirements identification. *SIGSOFT Softw. Eng. Notes* **7**(5), 88–93 (1982)
9. Davis, A.M.: Operational prototyping: a new development approach. *IEEE Softw.* **9**(5), 70–78 (1992)
10. Li, P.L., Kivett, R., Zhan, Z., Jeon, S.-e., Nagappan, N., Murphy, B., Ko, A.J.: Characterizing the differences between pre- and post- release versions of software. In: *33rd International Conference on Software Engineering (ICSE)*, pp. 716–725 (2011)
11. Coldrick, S., Lawson, C.P., Ivey, P.C., Lockwood, C.: A decision framework for R&D project selection. In: *IEEE International Engineering Management Conference (IEMC)*, vol. 1, pp. 413–418 (2002)
12. Bowers, J., Pycok, J.: Talking through design: requirements and resistance in cooperative prototyping. In: *SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pp. 299–305 (1994)
13. DeMarco, T.: All late projects are the same. *IEEE Softw.* **28**(6), 104 (2011)
14. Ferens, D.V.: The conundrum of software estimation models. *IEEE Aerosp. Electron. Syst. Mag.* **14**(3), 23–29 (1999)
15. Grimstad, S.: Understanding of estimation accuracy in software development projects. In: *11th IEEE International Software Metrics Symposium (METRICS)*, pp. 2–42 (2005)
16. Moloekken-Oestvold, K., Joergensen, M., Tanilkan, S.S., Gallis, H., Lien, A.C., Hove, S. W.: A survey on software estimation in the norwegian industry. In: *International Symposium on Software Metrics*, pp. 208–219 (2004)
17. Ellis, K., Berry, D.M.: Quantifying the impact of requirements definition and management process maturity on project outcome in large business application development. *Requirements Eng. J.* **18**(3), 223–249 (2013)
18. Damian, D., Chisan, J., Vaidyanathasamy, L., Pal, Y.: Requirements engineering and downstream software development: findings from a case study. *Empirical Softw. Eng.* **10**(3), 255–283 (2005)
19. Blyth, A.J.C., Chudge, J., Dobson, J.E., Strens, M.R.: A framework for modelling evolving requirements. In: *17th Annual International Computer Software and Applications Conference (COMPSAC)*, pp. 83–89 (1993)

20. Liou, Y.I., Chen, M.: Integrating group support systems, joint application development, and computer-aided software engineering for requirements specification. In: 26th Hawaii International Conference on System Sciences (HICSS), vol. 3, pp. 4–12 (1993)
21. Bowles, J.B.: Better software reliability by getting the requirements right. In: Annual Reliability and Maintainability Symposium (RAMS), pp. 110–115 (2006)
22. Moloekken-Oestvold, K., Joergensen, M.: A comparison of software project overruns — flexible versus sequential development models. *IEEE Trans. Software Eng.* **31**(9), 754–766 (2005)

Requirements Engineering for Internet of Things
4th Asia-Pacific Symposium, APRES 2017, Melaka,
Malaysia, November 9-10, 2017, Proceedings
Kamalrudin, M.; Ahmad, S.; Ikram, N. (Eds.)
2018, XIV, 205 p. 72 illus., Softcover
ISBN: 978-981-10-7795-1