The Impact of Product Development on the Lifecycle of Defects

Rudolf Ramler
Software Competence Center Hagenberg
Software Park 21
A-4232 Hagenberg, Austria
+43 7236 3343 872
rudolf.ramler@scch.at

ABSTRACT

This paper investigates the defects of a large embedded software development project over a period of about two years. It describes how software development and product development are organized in parallel branches. By mapping the defects reported on product development branches to the releases on the main line of software development, the paper shows the impact of the product development strategy on the defect lifecycle in software development.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management – *life cycle, software configuration management, software quality assurance (SQA)*.

General Terms

Management, Measurement.

Keywords

Product development, defect lifecycle.

1. INTRODUCTION

Software is the core part of more and more products. It is embedded in telecommunication equipment, medical instruments, household appliances, entertainment and multimedia devices, automotive components, and industrial machinery. While software is one of the critical success factors that determines the products' functionality and quality, it is the product as a whole that customers and users value. They may not even be aware about the software embedded in the products they buy and use. Hence, many companies have to embed software development into product development, and so their product strategy defines and constrains how the software is developed and evolved.

In this paper we investigate the development of a large embedded software system in the context of product development. The research objective of this paper is to measure the impact of the product development strategy on the defects in the software system. We therefore observe the major points in the lifecycle of every defect – the point were the defect is introduced, reported, and resolved in terms of the software release – and we trace back the defects reported in product development to the corresponding releases on the main line of software development.

The paper is structured to reflect the selected points of observation: Defect introduction, defect detection, and defect resolution in software development and in product development. Thus, after an overview of the studied project in Section 2, we discuss the three points of observation for the main line of software development in Section 3 and for product development in Section 4. In Section 5 we analyze the measured impact of product development on the main line of software development. Section 6 summarizes and concludes the paper.

2. PROJECT BACKGROUND

The observations in this paper are based on the analysis of the development of an embedded software system. The software system is the basis for a diverse range of products, which integrate the software system with hardware and peripherals from different manufacturers. Figure 1 depicts the software system's architecture. It consists of a runtime kernel and drivers to control the underlying hardware layer, as well as a number of applications providing functionality for the user. On top of the software system a generic user interface supports basic user interaction and the branding and customization of the products for different market segments and regions in the world.

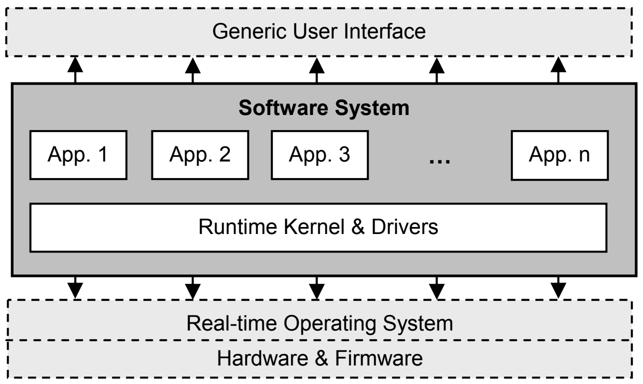

Figure 1: Software System Architecture

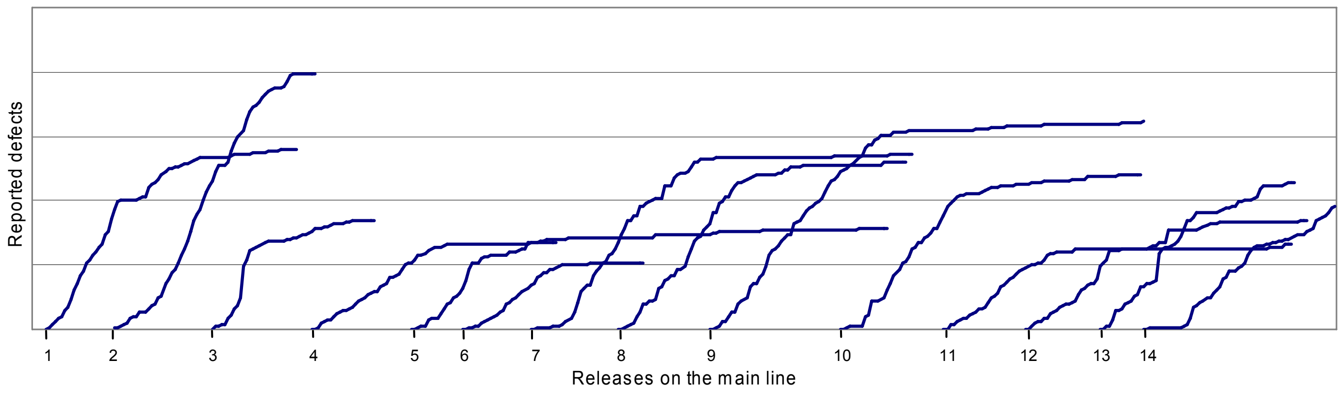

Figure 2: Accumulated number of defects for releases on the main line of development

The software system encompasses more than 700 KLOC C++ code. It has evolved to this size over three to four years of development with an increase of about 200 KLOC per year. In this time not only additional features were added but also new hardware platforms and peripherals were supported as well as various real-time operating systems.

Technically, software development and product development are organized using different branches in the version control system. Software development is conducted on the main line, products branch off from the main line and are evolved on parallel branches. In this study we investigated the evolution of the software system by analyzing the defects reported for about 130 releases of the software system and their impact on 14 consecutive releases on the main line (Figure 2), which capture about two years of ongoing development. The data used in this study is retrieved from the issue repository and the release database.

The **issue repository** contains more than 3.000 resolved issue reports. Besides detailed information about the nature of the issue, a report contains a link to the release in which the issue has been found. Issue reports concern either defects (68 percent) or enhancements (32 percent). In this study we investigate only defects, for which the report describes a failure of the software system and contains details about the fault in the code when resolved.

Release database: The evolution of the software system and the activities in context of product development are reconstructed from combining the data from the issue repository with the release database. The release database records about 130 releases in total, on the main line of software development and on the different branches. In addition, the release database associates the resolved issues to one or more releases in which the issue has been fixed, including the files that have been changed.

3. DEFECT LIFECYCLES IN THE MAIN LINE OF SOFTWARE DEVELOPMENT

The main line of software development proceeds in iterations of approximately eight weeks. Every iteration ends with the release of a labeled version on the main branch. Per iteration about 50 to 100 defects are reported and about the same number of defects are resolved. The majority of the defects are reported by the central testing department, which tests every release while development continues with the next release.

In the following we describe the three major phases of the defect lifecycle in the main line of software development.

3.1 Defect Detection

Defects detected on the main line of development are reported for releases on the main line. Figure 2 displays the cumulative defect occurrence over time for the 14 analyzed releases. The majority of the observable curves show a similar pattern, a small initial S-shape behavior. The number of reported defects varies and, furthermore, the majority of the curves overlap in time, since testing of one release may continue for several iterations. Yet, the resulting "short cycles" of defect detection in Figure 2 can clearly be attributed to the releases on the main line of development.

3.2 Defect Resolution

Defect reports are assigned to developers for resolution. In our study we only take defects into account for which a fix had been applied. Measuring the lifecycle of a defect from the release, in which the defect has been reported (R_r), to the release, in which the defect has been fixed (R_f) – see Figure 3 – shows, that about half (56 percent) of the defects are resolved within one iteration and the fixes are made available as part of the next release. 89 percent of all defects are fixed within 2 iterations. The remainder of the reported defects is fixed within at most 10 releases.

3.3 Defect Introduction

Due to the delay between the error, i.e., the event that led to the introduction of the defect, and the detection of the defect, the defect may be present in releases previous to the reported release (Figure 3). The release the defect has been introduced (R_i) is the first release containing the defect.

As an approximation for R_i, we traced back all reported defects along the line of releases by the following rule: *A defect is present in releases where all the files changed during a fix remain unchanged, going backward from the fixed release R_f, excluding the changes for the fix.* This rule has been optimized to avoid false positives, i.e. tracing a defect to a release that does not actually contain this defect, as the criterion for terminating the back-tracing is *a change to any of the involved files* although this change is not necessarily the cause of the defect.

For 21 percent of the defects reported on the main line of development it is possible to trace back the point they had been introduced to a release R_i before the reported release R_r. Computing the delay in number of iterations from introduction to reporting for defects where $R_i < R_r$ showed, that 52 percent of these defects were reported in the next release, 85 percent are reported within 3 iterations, and all defects are reported within at most 10 iterations.

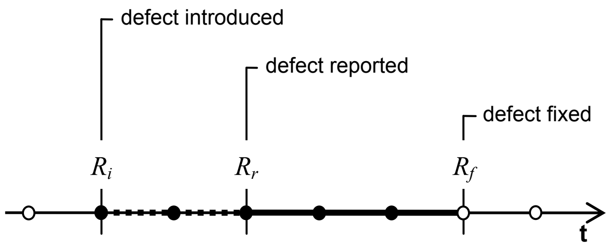

Figure 3: Releases containing defects (marked black)

4. DEFECT LIFECYCLE IN PRODUCT DEVELOPMENT

In this section we put software development in the context of product development. While the software system is developed on the *main line of development* (or main branch) as described in the previous section, software development in context of product development takes place in separate branches. On a *product branch*, a selected release of the main branch is advanced until it meets the product's requirements. *Integration branches* are used to integrate a stable release of the product branch with a product's target hardware. Figure 4 gives an overview of the different branches, which are described in the following.

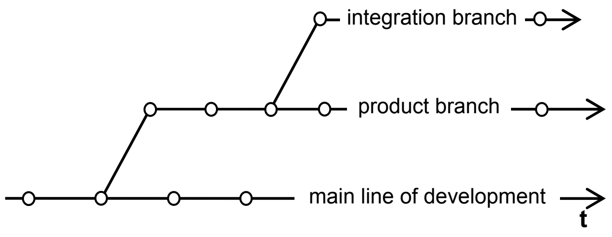

Figure 4: Branches in product development

Main line of development. The software system evolves along the main line of development. In every release new features are added, new hardware devices and new peripherals are supported, new user interface elements are implemented, and the integration with the real-time operation system is improved.

Product development, however, proceeds asynchronously from software development, usually at a faster pace, since the market is highly competitive and product lifecycles are short. Thus, product development does not accommodate for developing hardware or software from scratch. Instead, product development is dominated by the procurement and integration of third party components and the customization for different price segments and the needs of local markets. Often, a family of products is developed, based on a single platform but with different marketable features.

Product branches. Software development in the context of product development branches off from the main line when a consistent feature set and hardware support has been reached. On the product branch, the software system is stabilized and – although not intended but occasionally necessary – the feature set is evolved further until it meets the product requirements. Fixes and enhancements are merged back to the main line, where in the meantime development moves forward to the next generation of products.

Integration branches. When the software system is stable and complete, it is integrated with the hardware of the final product. Integration is a major step and requires close cooperation with product developers, hardware developers, and suppliers of third party components. It is therefore organized in a separate project and a separate integration branch is created. In integration projects, new releases of the software system are usually produced in weekly iterations. Often there are several integration projects and branches, one for every product of a product family.

Integration is also the last resort to find and fix defects. Once the product has been released on the market, it is usually not feasible to collect defect reports from end-users or to update the software integrated with the product. Thus, all relevant software defects have to be found and fixed before the product is released. Field defects do not play a major role for software development any more, even though they are carefully monitored and cared by product management.

4.1 Defect Detection

Over the analyzed period of two years of development, two product development cycles can be observed. Figure 5 depicts the number of defects reported per month over these two years for (a) the main line of development, (b) product branches, and (c) integration branches. The two product development cycles appear as two major peaks in the trend of the overall number of defects reported. The apparent two "long cycles" are a result of the overlying product development strategy, which defers testing and stabilization for product specific requirements until they become available.

At the time of the first release of the first product (marked as P1 in Figure 5) the number of defects reported for the main line of development rapidly dropped as many resources from development were shifted to support the stabilization on the product branch. Thus, the total number of reported defects still increased. Once the software system had been stabilized on the product branch, integration for the product started with relatively few additional defects reported and development on the main line regained speed.

For the second product (marked as P2 in Figure 5) additional resources where involved and, thus, development on the main line continued comparatively unaffected. Still, the total number of defects increased significantly due to the work on the product branch and – in this case, where several products of a product family had to be supported – also on the integration branches.

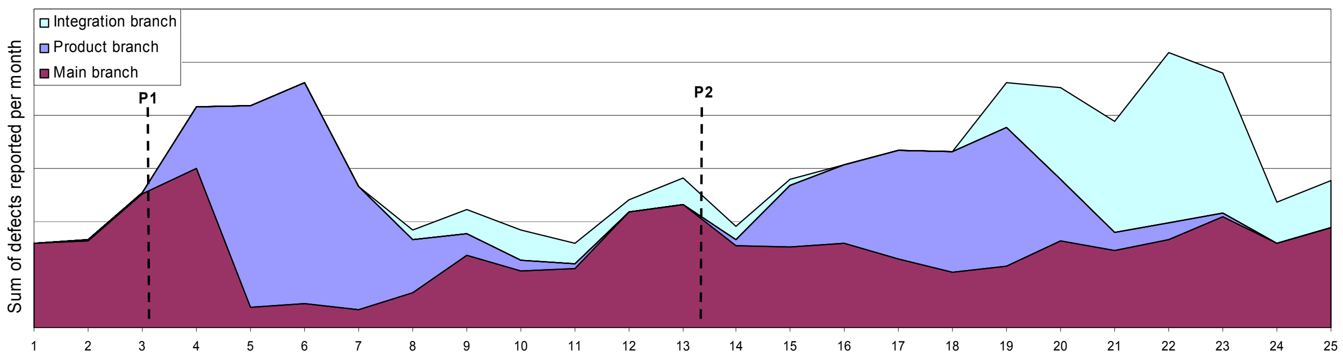

Figure 5: Defects reported on the main line, product branch, and integration branch per month

The main purpose of product and integration branches is the stabilization of the software system, i.e., finding and fixing of defects in the light of specific product requirements and hardware configurations. Hence, the defects reported on product and integration branches are detected in addition to what is continually found and fixed on the main line of development. On average, over the total time period observed, 45 percent of all defects were reported on product and integration branches. At the peak of the development of the first product, 90 percent of all defects reported were due to product and integration branches; 75 percent at the peak of the development of the second product.

4.2 Defect Resolution

In the studied project it is a strict policy that all changes have to be merged to the main line of development. Hence, development on the main line does not continue unaffected by the work on product and integration branches. Besides the fixes for the defects reported for a particular release on the main line, additional fixes are applied to the main line (R_f' in Figure 6) due to defects reported and fixed on parallel branches (R_f in Figure 6).

4.3 Defect Introduction

The defects reported on product and integration branches have either been introduced already in the main line of development, before branching, or in the course of stabilization due to a defective fix. As all changes are merged back to the main line, the defects reported on product and integration branches are present on the main line in any case.

The approach used to approximate the defective releases on the main line is illustrated in Figure 6. In a first step, the rule described in Section 3.3 was applied to trace the defect back on the product or integration branch, for which it had been reported, considering the files changed during the fix. When in tracing back the starting point of the branch had been reached, the rule was applied recursively for the release of the branch below, which had been used to create the branch. In a second step, if defect introduction had not yet been traced back to a release on the main branch, the defect was mapped to the chronologically next release on the main branch, making use of the fact that all changes are merged to the main branch. The so identified release R_i' is the earliest release on the main line containing the defect, unless an earlier release could be identified by tracing the defect back on the main line from release R_f'.

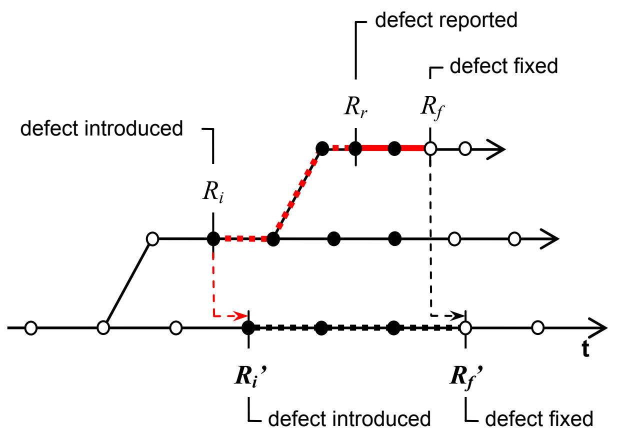

Figure 6: Mapping defects from branches to the main line

5. IMPACT AND IMPLICATIONS

In this section we show the impact, the defects reported on product and integration branches have on the releases on the main line of development. The impact is measured in terms of the increase in the number of defects introduced per release, the number of defects fixed per release, and the number of open defects per release on the main line. Our findings are summarized in Table 1.

The values given in the table are calculated as the percentage of the total number of introduced, reported, fixed or open defects due to defects reported on branches. In release 4, for example, 82 percent of the all defects introduced in that particular release were traced back from defects reported on a product or integration branches.

The first column of the table, **defects introduced**, shows an increase in these numbers for all releases. In half of the releases, the share of defect introductions found via reports on branches was even more than 50 percent.

For **defects reported** on branches there is no meaningful mapping to defects reported for releases on the main line. So, for this point in the defect lifecycle, there is no directly observable impact and the second column is therefore zero for all releases.

The third column, **defects fixed**, shows the number of fixes merged to a release on the main line due to defects reported on branches. For the first three releases there are simply no parallel or earlier branches, so no fixes had been merged. In seven of the eleven remaining releases, the share of defect fixes merged from branches is larger than 50 percent.

Defects open is the number of defects reported for this release plus the defects from earlier release, which have not yet been fixed. When analyzing the defects of a particular release, e.g., when constructing defect models based on the release's properties, the actual number of defects present, i.e. open, in this release is important. The last column in Table 1 shows the impact of defects reported on branches on the number of open defects. For example, in release 4 about two third (67 percent) of the open defects were reported on branches and mapped to the main line.

Table 1: Impact of defects from branches on the main line

Re-lease	Defects introduced	Defects reported	Defects fixed	Defects open
1	3%		0%	8%
2	5%		0%	8%
3	58%		0%	40%
4	82%		57%	67%
5	44%		80%	38%
6	23%		64%	24%
7	15%	0%	26%	19%
8	19%		3%	22%
9	23%		5%	28%
10	56%		12%	48%
11	72%		32%	62%
12	80%		57%	69%
13	63%		70%	67%
14	65%		70%	62%

Figure 7 further analyzes the observed increase in the number of open defects by comparing the number of open defects per release, showing both, the share resulting from the branches and from the main line. The distribution of open defects due to branches and main line can be explained by the product development cycles as depicted in Figure 5.

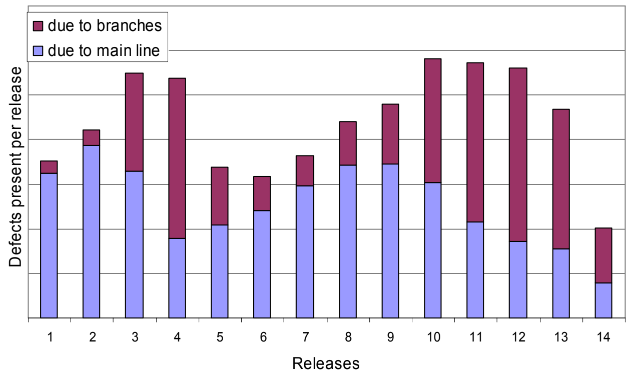

Figure 7: Impact of defects reported on branches on open defects per release on the main line

6. SUMMARY AND CONCLUSIONS

In this paper we investigated the defects of a large embedded software development project over a period of about two years. We described how software development and product development are organized as parallel branches, constituting the main line of software development, product development, and integration. By mapping the defects reported on branches to the releases on the main line, we were able to show the significant impact of the product development strategy on the lifecycle of the defects on the main line of software development.

Branching and merging in software development has been subject to several studies (e.g., [3, 4]) and some approaches and tools (e.g., [1, 5]) take the evolution of a software system across branches into account. The novelty in our work is the mapping of the defects reported on product and integration branches to releases on the main line of software development, including measuring the impact of defects reported on branches for the main line. Furthermore, in this paper we were able to explain the measured impact as a result of the pursued product development strategy.

From our observations and findings we can conclude that defects reported on branches have to be mapped to releases on the main line in order to obtain a complete and realistic status of the software system's defects. Our observations further imply that there is a latency caused by defect introduction on the main line and defect detection on branches during stabilization (bug-fix time [2]). Thus, when analyzing releases on the main line, the time frame has to be selected large enough to encompass all defects on branches which have an impact on the analyzed releases. We therefore currently study how long defects remain open to determine this latency and the corresponding impact of the product development strategy.

7. ACKNOWLEGEMENTS

This work was funded by the Austrian COMET Program. The author also thanks Prof. Giancarlo Succi and Prof. Barbara Russo from the Free University of Bolzano-Bozen for their support.

8. REFERENCES

[1] Fischer, M., Pinzger, M., and Gall, H. 2003 Populating a Release History Database from Version Control and Bug Tracking Systems. International Conference on Software Maintenance (ICSM'03), Amsterdam, Netherlands.

[2] Kim, S. and Whitehead, Jr., E. J. 2006 How Long Did It Take To Fix Bugs?. Workshop on Mining Software Repositories, ICSE 2006, Shanghai, China.

[3] Perry, D.E., Siy, H.P., and Votta, L.G. 1998. Parallel Changes in Large Scale Software Development: An Observational Case Study. International Conference on Software Engineering (ICSE'98), Kyoto, Japan.

[4] Williams C.C. and Spacco J.W. 2008. Branching and merging in the repository. Workshop on Mining Software Repositories, ICSE 2008, Leipzig, Germany.

[5] Zimmermann, T., P. Weibgerber, S. Diehl, and Zeller, A. 2004 Mining version histories to guide software changes. International Conference on Software Engineering (ICSE'04), Edinburgh, UK.