Optimizing Verification and Validation Activities for Software in the Space Industry

Robert Feldt, Bogdan Marculescu
Computer Science & Engineering
Chalmers University of Technology
Göteborg, Sweden
robert.feldt@chalmers.se

Jan Schulte, Richard Torkar
Blekinge Institute of Technology
Ronneby, Sweden
jan@janschulte.com

Philip Preissing, Erika Hult
RUAG Space AB
Göteborg, Sweden
philip@philippreissing.com,
erika.hult@ruag.com

Contact author (and presenter): Robert Feldt, robert.feldt@chalmers.se, phone: +46-733-580580, fax: +46-455-385057

Abstract—**Software for space applications has special requirements in terms of reliability and dependability and the verification & validation activities (VAs) of these systems often account for more than 50% of the development effort. The industry is also faced with political and market pressure to deliver software faster and cheaper. Thus new ways are needed to optimize these activities so that high quality can be retained even with reduced costs and effort. Here we present a framework for the management and optimization of verification & validation activities (VAMOS). An initial evaluation of the framework based on historical data as well as data extracted with a new tool has been done and are described briefly.**

I. Introduction

Software for space applications has special requirements in terms of reliability and dependability which requires the space industry to put a special focus on quality and standardization. The involved costs are huge, failures can be fatal and faults can be hard to fix after deployment. In recent years, the space industry is faced with rising political and market pressure to produce their products in shorter time and with less cost. The industry therefore aims at reducing their costs while keeping the same or achieving even a better quality of their products. For general software development, studies suggest that about 50% of total development costs account for quality assurance [1]. For quality-focused industries like the space industry these figures can often be even higher. Cost optimization of the verification and validation (V&V) process thus seems promising.

In an ongoing collaboration between Swedish universities and aerospace companies we have investigated ways to co-optimize cost and quality. In the first part of the work the focus was on analyzing the current situation in two case companies; how they work and the challenges they face regarding verification and validation [2]–[4]. The main problems found were: faults slip through between V&V activities, which means that faults are found in later stages than they could have been found, which in turn accounts for avoidable rework and thus higher costs; the selection of different V&V techniques as well as their interdependencies are not clearly understood; and the organizational costs that are related to compliance to space engineering standards are high.

In the second phase of the work a framework for managing and optimizing the verification and validation activities in space software development was developed [5]. Knowledge of the costs in relation to their benefits is necessary [6] to be able to minimize or improve those with no added value. Knowing the cost is also highly beneficial when speaking with stakeholders since the added cost can be traded off against benefits. There is also evidence that financial justification for software quality efforts makes them easier to justify and gives a better basis for implementing them [7].

This paper summarizes the work on the framework and also introduces a new tool that has been developed to extract metrics about the effectiveness of code inspections.

II. Background and Case Company

A. RUAG Space AB

RUAG Space AB (RUAG) was formerly known as SAAB Space AB but was then acquired by RUAG Space. RUAG has a very long experience in the design, development and delivery of both hardware and software for computer and data-handling products for space applications. The main product areas are Data management systems, Fault-tolerant computers and processor products, Payload control computers, Data processing, and Small mass memories. The software developed by RUAG for these computers is in the range from small boot software to full application software, but the main focus is on hardware-near, embedded, real-time software. The software development process used is based on the ECSS standards, mixed with an integration driven development approach. The company is headquartered in Göteborg, Sweden and employs in total 360 people, of which about 30 work in the software unit. Typically up to five projects are developed simultaneously in varying team sizes of about 10 people. The software is developed mainly in C but with some low-level parts written in assembler.

A management and optimization framework for verification and validation activities called VAMOS was recently created in relation to RUAGs development process [5]. In VAMOS the cost reducing effort is essential, however to actually be able to measure and reduce costs the approach needs to be complemented [6]. This paper summarizes VAMOS and presents a new tool that is based on the framework.

B. Related work

Instead of focusing on a single verification and validation activity, both research [8], [9] and industry experience [10] show that combining different VAs can be more efficient in finding defects. To keep up with the constant market pressure that is omnipresent in industry, further investigation is necessary to guide industry on how to select VAs to maximize the defect detection efficiency while minimizing the effort spent. Some initial work that has been done towards the creation of such frameworks is presented in the following.

Wojcicki and Strooper present an iterative selection strategy (ISS) for verification & validation activities [11]. The VAs are thereby firstly selected by their efficiency in order to maximize completeness, and secondly by the effort, i.e. the cost, they require. Based on the data collected while applying the technique, the selection is refined iteratively.

In [12] the concept of Fault-Slip-Through (FST) is presented. In this approach, the faults found are categorized according to which phase they belong to, i.e. in which VA they should have been found. Based on these findings and the effort of the VA, the improvement potential for each activity is calculated. In contrast to classic fault measurement techniques like phase containment metrics [13], faults-slip-through considers the most cost-effective phase to find a defect.

Wagner [14] proposes a more analytical approach. The defect detection techniques are thereby compared using an economical metric, namely the return on investment (ROI). The model also considers the effect of combining different defect detection techniques.

A framework for the comparison of testing activities and formal verification is presented in [15]. However, this approach focuses more on studying the synergy and relationship of these two activities and doesnt give any advice on how a combination of them can be optimized.

Apart from the work that focuses on optimizing VAs in particular, frameworks have been proposed like QIP [16] which target process improvement in general. They typically follow an iterative process based on a problem description or analysis, a measurement phase, a root-cause analysis followed by an implementation phase.

III. VAMOS

Figure 1 gives an overview of the VAMOS framework with its five steps (Define, Measure, Analyze, Improve, Implement) and two supporting quality gates/models (ADC and MOM). Four of the steps are used in consecutive iterations of the development. As a pre-cursor the Define step details what are the overall aims of using VAMOS. In the first iterative step, the necessary data is *measured*. This data is then *analyzed* to find out the problems in the VAs. The analysis is based on FST concepts. In the *improvement* phase, adequate changes are developed to address the identified problems. A static evaluation of the changes are done with respect to their predicted benefit. In the last step, these changes are *implemented* in the development process. Since this is an iterative cycle, the actual benefits of the change will be measured in the next iteration and thus, can be directly evaluated.

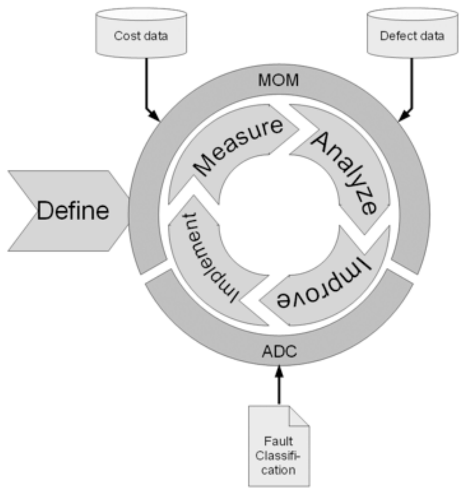

Fig. 1: Overview of the VAMOS framework

Since VAMOS is based on the FST [12] and the ISS [11], it requires similar inputs. These are the defect data, i.e. how many faults are found by each VA, and the cost data, i.e. how much does it cost to perform the VA and how much does it cost to fix a defect. Furthermore, a fault classification is needed which is used to classify the defects and facilitate an analysis of the root causes of the identified defects.

To make sure existing measurements meet the necessary quality to be reused in the framework and to facilitate the development of new measurement processes for VAs or companies that do not yet have one, two quality gates surround the framework. The Measurement Options Model (MOM), is used to assure the quality of the cost and defect data but also to ensure that as little data are collected as possible and that it can be collected as efficiently as possible. The Adaptive Defect Classification (ADC) is used to guarantee the fault classification is usable for the framework or, if it is not, to help develop a new fault classification which is adapted to the company and the domain the company is working in. Together, the MOM and ADC ensures that VAMOS can be adapted to specifics of individual organizations; they thus help generalize the framework.

IV. Initial Evaluation of VAMOS

VAMOS and its components have been evaluated in a number of steps as described below.

A. *Company-specific Defect Classification Scheme*

Before conducting this study, the company did not use a consistent and internally developed defect classification scheme. Defects were either not classified or classified using the default classification scheme from the vendor of the bug-tracker in use. This scheme was not adapted to the companys domain or the VAMOS framework. Hence, a defect classification had to be developed that was tailored for the framework and the domain of embedded aerospace applications. Therefore, the fault classification was created from scratch based on the ADC.

The developed fault classification was reviewed in a meeting with four representatives from RUAG. The people were chosen so that a responsible and experienced person for every VA was present. The meeting resulted in several minor changes as well as a change in priorities of classes, but no changes to the fault classes themselves.

The final classification (major categories) was:

- HW interface
- Timing/Concurrency
- Algorithm / Interface
- Robustness
- Understandability
- Beautification

This is notably simpler than if a standard solution like ODC would have been used [17]. This simplicity results in a simpler framework which is more likely to be used and thus give the type of benefits that are aimed for.

B. *Mesurement Options*

An extensive quality analysis of the currently available data sources at RUAG revealed that no measurement fully satisfies the demanded quality requirements. Hence, no measurement may be used without modification in VAMOS. Therefore, a measurement options catalog was developed to derive the measurements for each VA. Each measurement option for each VA was then rated according to its fixed effort, variable effort and accuracy. Additionally, it was evaluated how to perform measurement of the removal of defects and of regression effects. The list was then sorted and filtered according to the process to achieve the final list of measurement options for each VA.

The measurement options catalogue and the rating of them were performed by the developers. This was done to avoid having several meetings with responsible persons from RUAG and to interfere too much with their daily work. Since the measurement options may require changes in the processes, it was however not possible for the researchers to decide whether tools and documents may be changed and how much effort can be spend on the measurements. Therefore it was decided to hold a workshop with the head of software development and two project managers to verify the measurement options, discuss alternatives, get additional feedback and possibly propose other measurement options. The measurement options were in general perceived positively and only small changes and improvements were proposed.

C. *Actual Evaluation*

The evaluation was performed on historical data from a finished project. Here we only discuss the evaluation and its results from a subset of the investigated process phases.

A design review was performed by three different reviewers. The historic analysis was done by evaluating the defects reported in these three review documents. It revealed that most faults were of the types Understandability and Beautification.

For code inspection most defects that had been found were Understandability defects. Additionally a lot of Beautification defects are found. Almost none of these faults had slipped through (in the FST sense), i.e. almost all understandability and beautification faults could only have been found in CI. This usually means that no problem exists. However, because of the large number of defects it might be desirable to reduce this number of defects by developing means to prevent these defects from occurring. More important and interesting is that a large number of fault slip-through was seen in interface faults from the design review to the code inspection. This indicates that the design review needs to be improved. Another significant fault slip through exists from the unit testing for algorithm and robustness defects.

Based on these, and similar, VAMOS steps the final analysis revealed that the design review had the highest improvement potential for interface faults. Thus, the interface faults from the code inspection sheets were further analyzed by doing a root-cause analysis.

This resulted in the conclusion that the ‘missing interface description’ defects can be prevented from reappearing by having automatic checks for empty descriptions in the design documents. The ‘unnecessary operations or constants’ defects can similarly be checked by an automated tool. Based on cost data the actual number of hours to be saved by these improvements could be calculated. Thus any effort and costs can be traded off against future benefits.

V. Data extraction from code inspection logs

Code inspection is essentially a qualitative process that results in a certain code passing or failing on a number of points that are of interest. These points of interest that the code must pass before clearing the code inspection are hereafter called checkpoints. The checkpoints describe certain standards that code must fulfill in order to conform to the quality requirements of the company. A code inspection sheet gives qualitative information regarding whether or not that code passed the inspection with respect to the checkpoints mentioned above.

The method code inspections are held at the moment, however, does not allow any quantifying information to be centralized for a given project. While all code inspection sheets contain the same checkpoints, there is no way to clearly and efficiently determined which of these checkpoints cause the rejection of certain code and, as a result, which kind of fault is more commonly found during the inspection. To deal with this aspect, an internal tool has been developed. The purpose of this tool is to identify, for a given project, what checkpoints were used in the code inspection sheets and how many times, i.e. for how many documents, were these checkpoint passed, failed or not applicable.

This is achieved by analyzing all the code inspection sheets and comparing the checkpoint against a library of checkpoints encountered so far. Such a design enables further extensions, should they be needed. One example of such an extension is the automatic generation of code inspection sheet templates by selecting the required checkpoints from a library. This would reduce the number of cases where several such checkpoints have different wording but a basically similar meaning.

The purpose of this tool is to offer an overall view of the code inspection documents resulting from a project. The tool enables the user to get information regarding which checkpoints have failed more often and how many times such failures have occurred. Information is also stored regarding how many times the checkpoint has been passed and how many times it was deemed not applicable, since projects may contain more or less code, any attempt at analysis must take the relative number of failures into consideration rather than the absolute number.

VAMOS is mostly concerned with the most efficient and effective ways of finding faults, since the purpose is to minimize the cost associated with the detection and repair of faults. RUAG has identified in a previous study code inspection as being the most efficient way of finding faults within the software. This evaluation, however, cannot be proven until some measurements are introduced that enable such comparisons to be made between different validation activities. This tool is a first step towards that objective. By creating some measurement of the faults found with this particular validation activity, a baseline is created against which further work can be measured. The ability to add quantitative information to the available qualitative information is the starting point for any attempt at introducing measurements based comparisons between validation activities.

The tool is designed to handle any type of validation activity that relies on the passing of certain checkpoints, and thus is not limited to code inspection sheet analysis. As such it can form a basis for a comparison between any validation activities that fulfill that description. Since the focus of VAMOS is to compare validation activities based on strict metrics, this tools offers the starting point of such an evaluation.

VI. Discussion

During our evaluation it was uncertain whether an application of VAMOS on historic data could be performed. However, in the end performing it even on incomplete data proved to be very valuable. It helped to better judge the effort of applying the framework and to find practical problems in the process steps. Furthermore, it showed the benefit of VAMOS and also aided in communicating it among the organization, because people have a connection to the historic project and are therefore much more interested in the results. It is also interesting to compare the results of the dynamic validation to the implicit assumptions of RUAG about their process, e.g. which activities are effective and which are not.

For the quality of results, it would of course be the best to measure every single data item. However, in industrial contexts this is not possible because of the overhead associated with it. Therefore, different measurement levels have to be supported: from direct measurement over sampling to estimations or even not measuring a particular figure at all. While this, especially for researchers, might often seem unscientific, there is no way around when developing concepts to be applied in industry. Instead, the companies should be assisted in doing their estimations and samplings in the best way to achieve a maximum accuracy. The MOM is a promising effort in this direction. The fairly simple and limited measures it resulted in is a benefit compared to more general process improvement solutions.

Since VAMOS is based on FST, it takes into account cost considerations. Therefore, it does not aim for just improving any VA, but for improving the VAs in the most cost-effective way. Additionally, it helps to find out which VAs do not find the faults they are supposed to find.

Furthermore, VAMOS provides an overlap factor to analyze whether two different VAs find the same kind of faults and it is more cost-efficient to remove one of the VAs (either completely or only for a certain fault type). Although, it may not be possible to remove a VA because of standards that has to be followed, this allows a company to show to its customers what the different VAs cost and what the cost for quality is.

Therefore, VAMOS combines the benefits of the ISS, i.e. the optimization of the selection of VAs, with the benefits of the FST, i.e. the optimization based on the faults that cost the most. As mentioned before, VAMOS provides concrete recommendations on measurements, analysis and improvements. In contrast to FST, VAMOS defines ways to derive the improvements from the analysis. Improving VAs to find defects earlier leads also to a better quality of intermediate releases, which in turn improves customer satisfaction and lowers the amount of support a company has to provide during these releases.

The biggest drawback of VAMOS is that it requires several measurements, which is not only true for VAMOS, but for any process improvement framework that relies on empirical data. To limit this overhead, the MOM aims for measurements that require a low effort and different measurement levels exist.

Although VAMOS incorporates cost-considerations, it does not take into account intangible benefits. However, VAMOS also supports the calculation of other figures such as the return on investment (ROI), which can incorporate these benefits if they can be expressed as a monetary value.

A further drawback is that the framework requires that the developer classifies the defects according to the VA where they could have been found and the fault type. Since this can result in errors, it may be necessary to adjust the fault types and train the developers.

VII. Conclusion

This paper presented a framework for the management and optimization of verification & validation activities (VAMOS) which was developed at the RUAG Aerospace Sweden AB in Göteborg. The framework allows for an evaluation and comparison of VAs according to their efficiency and effectiveness and furthermore for a process improvement that takes into account economic aspects. No such framework currently exists and previous work is either targeted at general process improvement or solves only parts of the problems. Specifically, it expands the concept of FST to an iterative framework that also defines how to derive improvements and calculates the benefit of them. Furthermore it enhances the ISS by not only trying to have a minimal set of VAs, but also by trying to improve existing VAs.

VAMOS follows an iterative approach similar to Six Sigma or QIP and is furthermore based on the concept of fault-slip-through and an existing iterative selection strategy for verification activities. Thus it combines the benefits of FST, i.e. to take into account cost-considerations to improve only the VAs with respect to the most expensive defects, with the benefits of ISS, i.e. to find the optimal and most cost-efficient combination of VAs.

Using data of an historic project at RUAG, the framework was evaluated by performing a first iteration of it on this data. The results showed that VAMOS can reveal the problems in the process and deliver useful improvement results. In the case study problems in the design review were discovered with VAMOS which were also confirmed by the software development team in a team meeting. A tool to extract metrics from code inspection logs was later developed to further help in verifying VAMOS. The script helped find checks in code inspections that frequently find defects or are not applicable. Future work will involve evaluating VAMOS more realistically in ongoing projects.

Acknowledgment

This work has been supported by the Swedish National Space Board under the 'National Space Research Project'.

References

[1] G. Myers, T. Badgett, T. Thomas, and C. Sandler, *The art of software testing*. Wiley; 2nd edition, June 2004.

[2] R. Feldt, R. Torkar, E. Ahmad, and B. Raza, "Challenges With Software Verification and Validation Activities in the Space Industry," in *3rd International Conference on Software Testing, Verification and Validation (ICST 2010)*, April 2010.

[3] R. Feldt, E. Ahmad, B. Raza, E. Hult, and T. Nordebäck, "Evolving the ECSS standards and their Use: Experience based on Industrial Case Studies," in *Data Systems in Aerospace 2009 (DASIA'09)*, 2009.

[4] E. Ahmad and B. Raza, "Challenges with Software Verification and Validation in the Space Industry Constrained by the ECSS Standard," Blekinge Institute of Technology, Tech. Rep. MSE-2009:02, February 2009.

[5] J. Schulte, "A Software Verification & Validation Management Framework for the Space Industry," Master's thesis, Blekinge Insitute of Technology, Ronneby, Sweden, September 2009.

[6] P. Preissing, "A Framework for Improving the Software Verification and Validation Process in the Space Industry," Master's thesis, Technische Universitt Munich, Germany, September 2009.

[7] S. Slaughter, D. Harter, and M. Krishnan, "Evaluating the cost of software quality," 1998.

[8] B. Littlewood, P. Popov, L. Strigini, and N. Shryane, "Modeling the effects of combining diverse software fault detection techniques," *IEEE Transactions on Software Engineering*, vol. 26, no. 12, pp. 1157–1167, 2000.

[9] B. Kitchenham and S. Linkman, "Validation, verification, and testing: diversity rules," *IEEE software*, vol. 15, no. 4, pp. 46–49, 1998.

[10] N. Kikuchi and T. Kikuno, "Improving the testing process by program static analysis," in *Software Engineering Conference, 2001. APSEC 2001. Eighth Asia-Pacific*, 2001, pp. 195–201.

[11] M. A. Wojcicki and P. Strooper, "An iterative empirical strategy for the systematic selection of a combination of verification and validation technologies," in *WoSQ '07: Proceedings of the 5th International Workshop on Software Quality*. Washington, DC, USA: IEEE Computer Society, 2007, p. 9.

[12] L.-O. Damm, L. Lundberg, and C. Wohlin, "Faults-slip-through - a concept for measuring the efficiency of the test process," *Software Process: Improvement and Practice*, vol. 11, no. 1, pp. 47–59, 2006.

[13] A. Hevner, "Phase containment metrics for software quality improvement," *Information and Software Technology*, vol. 39, no. 13, pp. 867–877, 1997.

[14] S. Wagner, "A model and sensitivity analysis of the quality economics of defect-detection techniques," in *Proceedings of the 2006 international symposium on Software testing and analysis*. ACM, 2006, p. 84.

[15] J. Bradbury, J. Cordy, and J. Dingel, "An empirical framework for comparing effectiveness of testing and property-based formal analysis," *ACM SIGSOFT Software Engineering Notes*, vol. 31, no. 1, p. 5, 2006.

[16] V. Basili, G. Caldiera, and H. Rombach, "Experience factory," *Encyclopedia of software engineering*, vol. 1, pp. 469–476, 1994.

[17] R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, D. S. Moebus, B. K. Ray, and M.-Y. Wong, "Orthogonal defect classification-a concept for in-process measurements," *IEEE Trans. Softw. Eng.*, vol. 18, no. 11, pp. 943–956, 1992.