Information Needs for SAFe Teams and Release Train Management: A Design Science Research Study

Miroslaw Staron[1], Wilhelm Meding[2], and Poupak Baniasad[3]

[1] Chalmers | University of Gothenburg, Gothenburg, Sweden
miroslaw.staron@gu.se
[2] Ericsson, Sweden
wilhelm.meding@ericsson.com
[3] Software Center, Sweden
poupak.baniasad@software-center.se

Abstract. Large, embedded software development companies increasingly often transform from V-model development to Agile practices, as they want to increase their customer responsiveness. Their teams transform and they are faced with the new challenges on how to measure progress, quality and scope over time. Their management evolve and need new kinds of dashboards to address their information need and ease decision formulation processes. The goal of this paper is to identify the set of measures and indicators important for Agile embedded software development based on SAFe. We studied a large automotive company and identified the information needs of their teams and SAFe release train management. The results show that the three main areas to monitor are: scope creep, defects carried over to integration and integration status. Based on the results of our work, some of the elicited information needs and measures were implemented in forms of dashboards (which we present in the paper). By comparing to the existing literature, we concluded that the set of measures prescribed by SAFe is not sufficient in practice and needs measures relating to scope creep, defects carried over to integration and integration status.

1 Introduction

Agile software development gained high popularity in different software development domains, starting from web development and now becoming increasingly popular even in the embedded systems domain. One of the modern achievements is continuous software integration and deployment. They aim to improve the quality of software products and their availability to the market [2] by shortening feedback cycles and providing customers with as up-to-date software as possible. However, they require software development to progress at a higher speed than before and work in ecosystems of software development organizations [3]. In order to achieve this higher speed, companies focus on customer

data analytics and optimization of software development towards faster deliveries. Agile metrics are an important part of that work as they provide insight into the progress and quality of software product development; they also provide a means of communication within Agile organizations [4].

Automotive software companies are no exception, although their context is more challenging than, for example, the Web 2.0 companies, since:

- their software needs to fulfill safety critical requirements (e.g. ISO/IEC 26262 standard [6]),
- their products are often part of a complex ecosystems of suppliers (process ecosystem) and technologies (software ecosystem) [13], and
- the lifecycle of their software is over 10 years as the software is usually organized in form of platforms that support different product lines on each platform.

These challenges resulted in the development of Agile-based software development methods for these companies. SAFe [7], Scaled Agile Framework for Lean Software and Systems Engineering, is one of such frameworks. The framework prescribes a number of measures (e.g. velocity planned, unit test coverage), related to the Agile ways-of-working. On the other hand, standards like ISO/IEC 26262 prescribe measures that seem to oppose to the empowerment of teams encouraged by the Agile ways-of-working, e.g. measuring the progress of formal verification of the release-ready software.

Therefore, we set off to study the practice of applying SAFe, in particular we explored what the information needs of Agile software organizations in the embedded software development are. In this paper, we report on the results of studying one large Swedish automotive OEM. We address the research question:

What are the information needs of SAFe teams and train management in the automotive domain?

Our work follows the design science research methodology [18]. The study is based on workshops with different stakeholders at the company, including software development teams, software product management, line management, quality management and train management. Our study was conducted over a period of nine months and concluded with the development and introduction of a number of dashboards at the company which address the information needs that we found.

The results show that the main information needs are:

- scope creep – new functionality added to the team's backlog after the start of a sprint or program increment,
- defects carried over into integration – defects which are not discovered in earlier phases and result in decreasing the speed of integration, and
- integration status – availability of test equipment, stability of builds and defect turnaround time.

Based on the results, we concluded that the standard, prescribed, measures of SAFe do not address these information needs sufficiently. They do not provide

the necessary insight for the teams and the management. Therefore, in practice, new measures,indicators and visualization dashboards need to be added.

This paper is structured as follows. Section 2 presents an overview of the work in SAFe process management, dashboards for Agile teams in general and automotive software measurement. Section 3 presents the measures that are prescribed by SAFe. Section 4 presents the design of our study and Section 5 describes the results. Finally, Section 6 and Section 7 discuss the validity of our study and our conclusions.

2 Related work

Measures for Agile software development have been studied in several cases, and the main consensus is that the standard measures must be complemented with the domain specific ones. For example, Meding [9] studied Agile teams and identified the most important measures for their Agile teams. These measures were a combination of Agile measures (e.g. backlog) and domain specific measures (e.g. number of defects reported by customers). Schermann [12] came to similar conclusions by conducting a meta-analysis of previous studies. They recognized the need to balance confidence and velocity when monitoring Agile software development.

In order to find this kind of combination of measures, we can explore the measures for DevOps, as they combine the development and operation measures. An example of a study on DevOps, which has been done this in the domain of development of finance systems, has been done by Huijgens et al. [5]. Although that study has identified a number of important metrics (e.g. cycle time), the metrics are related to project progress and not product development or product features. Even in the same domain, i.e. embedded software development, there is not much more than the standard metrics for tracking progress, e.g. a study at ABB by Augustine et al. [1]. Studies of wider set of projects also seem to focus on non-product related metrics, e.g. Ostakhov et al. [11].

On the other hand, there are studies that show that standard dashboards are not sufficient, e.g. Liechti [8]. The information provided by dashboards is always pre-defined, both for project-oriented dashboard of progress monitoring and product-oriented dashboards for code quality. This pre-definition requires complementing the dashboards with qualitative data from customer meetings, reviews, and similar fora.

3 Theory: Measures recommended by SAFe

SAFe development methodology recommends a number of measures, categorized in four areas: lean portfolio, program, large solution, team. The full set of measures is available in the process documentation at [7] Table 1. Within each category, the SAFe framework also defines a number of areas. For the sake of space, we provide the most relevant measures for our work.

Table 1. Relevant Portfolio measures

Area	Measure	Measurable concept
Lean portfolio	HR Statistics	Employee satisfaction
Lean portfolio	Net promoter score	Customer satisfaction
Lean portfolio	Feature cycle time	Productivity
Lean portfolio	Team, program, large solution and portfolio self-assessment	Improvement
Lean portfolio	Release predicability	Improvement
Lean portfolio	Support call volume	Quality
Lean portfolio and Enterprise balance scorecard	Number of releases per year	Time to market
Lean portfolio and Enterprise balance scorecard	Number of defects	Quality
Enterprise balance scorecard	Team velocity vs. capacity	Efficiency
Enterprise balance scorecard	Teamwork	Agility
Enterprise balance scorecard	Value feature point delivered	Value delivery

The most important area for our work is from all areas of the category of team measures and from the solution train performance (STP) area of the large solution category. They are presented in Table 2

The measures which we assessed as less relevant were measures related to innovation accounting (e.g. number of customer visits on site), self-assessments (e.g. stakeholder engagement), and pipiline efficiency (e.g. validation on staging).

During the problem awareness identification phase, we understood that these measures may not be sufficient, as in our previous work, agile teams in other companies were asking for more product-oriented measures (e.g. architecture stability [13]), more detailed project-oriented measures (e.g. defect backlogs [9], [14]) or feature flows [15]. Therefore, we designed a set of dashboards to help SAFe organizations to monitor their software products and software development beyond the basic concepts of velocity or number of new test cases.

4 Research design

We use the design science research method to understand the information needs, to design the dashboards (mock-ups) and to evaluate these mock-ups. In this study, we set off to explore the research question of *What are the information needs of SAFe teams and train management in the automotive domain?*

4.1 Problem awareness, case and subject selection

In order to address the question, we selected a case company which is an automotive OEM from Sweden. The OEM is developing software both in-house

Table 2. Relevant team and large solution measures

Area	Measure	Measurable concept
STP	Program velocity	Functionality
STP	Predicability	Functionality
STP	Number of features planned	Functionality
STP	Number of features accepted	Functionality
STP	Number of enabler features planned	Functionality
STP	Number of enabler features accepted	Functionality
STP	Number of non-functional tests	Quality
STP and team	Number of stories planned	Functionality
STP and team	Number of stories accepted	Functionality
STP and team	Unit test coverage	Quality
STP and team	Number of defects	Quality
STP and team	Number of total tests	Quality
STP and team	Percent of automated test	Quality
Team	Perecent of stories accepted	Functionality
Team	Velocity planned	Functionality
Team	Velocity actual	Functionality
Team	Number of new test cases	Quality
Team	Number of new test cases automated	Quality
Team	Number of refactors	Quality

and through suppliers, which is a standard way for this market. The OEM has transformed its operations from the classical, automotive V-model based development [14] to the modern SAFe development in the past two years. The scope of the transformation was the entire product development program, but in this paper, we focus only on the software development part. The scope of the software development is still significant, with over 100 developers affected in the OEM and in its suppliers.

The studied company was large, which meant that there were several roles involved in software development, integration, testing, deployment and management. Therefore our study was based on a number of workshops with different stakeholders. We selected all stakeholders based on their experience and a specific information need that they could have (which we elicited beforehand through discussions conducted at the company by one of the authors).

To build awareness of the problem with the existing measures and to elicit information needs from the stakeholders, we conducted a series of workshops.

Workshop 1: All stakeholders First we conducted a workshop with all stakeholders, in total 22 persons, with the roles representing: software product managers, software designers, software teams, quality management and release management. The goal of the workshop was to understand the diversity of the information needs. The main method for data collection was the so-called brainwriting, where each participant prepared a set of post-it notes about their information needs and then presented them to the entire group. In this way,

we could avoid the problems of dominating the discussion by the most active participants.

During the workshop, we also grouped the information needs and evaluated the groups with the workshop participants. We documented the results by taking notes of the discussions (one of the authors) and by noting the groups of information needs elicited by the workshop.

Towards the end of the workshop, we also agreed on which group of information needs should be discussed next and which group of stakeholders was the most relevant for that discussion. We identified which stakeholders should be invited to the subsequent workshops.

Thematic workshops We conducted six thematic workshops with groups of stakeholders, focused on one specific area identified in Workshop 1. Each thematic workshop was conducted with a smaller number of stakeholders, in order to capture the information needs of the right stakholders and not to discuss issues that are out-of-scope for that particular reason. The criteria for selecting the stakeholders were that the stakeholder should:

- have the mandate to react upon the measures (e.g. be line manager responsible for integration),
- have the ability to make change (e.g. assign extra resources, reduce scope), and
- be recognized as the responsible for that particular area of interest (e.g. release train).

SAFe train management (also referred to as Program Manager) was represented by one stakeholder, who had a number of years of experience with working within the organization. He has been working both before and after the transformation to SAFe. The development team was represented by two stakeholders, one of them was an experience engineer working for a number of years at the company, and the second was a junior engineer with experience with software measurement.

The integration and test team was represented by three persons who had similar background to the Program Manager, in particular they worked at the company before and after the transformation from the V-model to SAFe.

4.2 Prototype development

In order to organize the work with prototype development, we used the process of developing measurement systems from [17] and [16]. The process is based on the international ISO/IEC standard 15939 [10], which defines the notions of information need, stakeholder and measure [14]. Conceptually, the elements (different kinds of measures) which are used in the measurement process can be presented as in Figure 1.

The model provided a framework to organize the collected data and to organize the results into dashboards and measurement systems, which were used at the company.

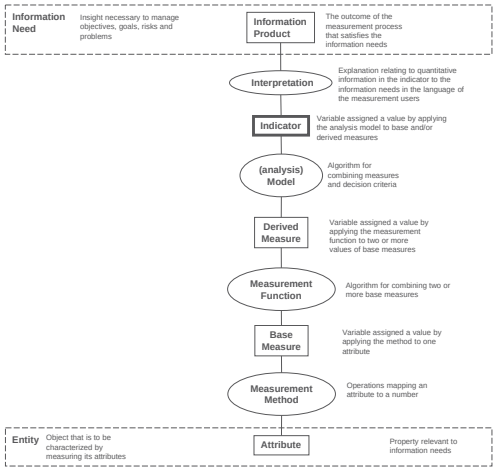

Fig. 1. Measurement system information model (from ISO/IEC 15939:2007)

4.3 Prototype evaluation and analysis methods

The analysis methods for all of the evaluations was that one person was conducting the evaluation and two researchers were taking notes. Then one of the researchers compared the notes and summarized the results.

In the workshop with all stakeholders, we used post-it notes and thematic groupings as the analysis methods. For the thematic workshops, in the analysis of notes we used thematic analysis, where we used the results of the first workshop as the themes.

Each dashboard was evaluated by the stakeholder. We presented the dashboards and the stakeholder was asked to assess whether it fulfilled his/her information needs. If the needs were not fulfilled, a change request was made. An example of such a case was wrong filtering criteria in one of the dashboards – instead of showing test progress since the last test run, the dashboard showed the cumulative one. The dashboards were also presented for other companies during a workshop (10 companies present ranging from medium to large size, all developing embedded systems and all using variations of Agile software development).

5 Results: Information needs and dashboards

Based on the first workshop, we identified three categories of information needs:

– Scope creep: in large organizations there is a tendency to understand Agile as a flexible way of working and lack of planning, instead of the flexibility to plan the sprints and do not change the scope within the sprints. Therefore, in the workshop we identified the need to quantify the work that is added to the team's backlog during an ongoing sprint.

– Defects into integration: for safety critical systems there is a need for stringent integration and testing phases, in order to secure the safety of the software product. Therefore a high quality software which is integrated and tested is important for the continuous delivery and deployment; if the software does not meet quality requirements, defects are reported and removed, which slows down software development.
– Integration status: in distributed organizations, the teams can deliver software many times during the day and it is important that they can do that, in particular it is important that the continuous integration toolchain is working fast and without problems. Therefore, we identified the need to monitor the availability and speed of tools used for integration.

In this section we group the results from all workshops per category above and present them per category.

5.1 Scope creep

We identified two stakeholders who have the interest, mandate and ability to react upon the indicators of scope creep: product manager (ProdMan) and development team (DevTeam). Their information needs are partially complementary.

The results are presented in Table 3.

Table 3. Relevant team and large solution measures

Stakeholder	**Information Need**	**Measure(s)**
ProdMan, DevTeam	What is the status of our field tests?	Number of Problem Reports (PR) from the field tests
ProdMan, DevTeam	What is our status of problems from the previous scope (from customer)	Number of PRs from customer
ProdMan, DevTeam	What is our status of legacy defects?	Number of internal PRs (previous release defects)
ProdMan, DevTeam	What is our planning accuracy?	Difference between estimated development time and actual development time
ProdMan, DevTeam	How much do we support other teams?	Number of resolved issues (internal defects) + burn-up + fluctuations in velocity
ProdMan, DevTeam	How much do we support other teams?	Number of new issues in internal defect reporting tool
DevTeam	How many changes in the development environment do we experience?	Number of changes introduced to tooling
DevTeam	What is the status of our quality journal?	Number of open problem reports
DevTeam	How dependent are we on other teams?	Expert assessment

We found that the company provided the teams with the support for different tools for categorizing different types of defects, which are called problem reports at the company: one for team's defects that need to be resolved within the development organization, another for defects reported by other organizations (e.g. testers), and the third for the defects reported by the customers and from field testing. The different phases, where the defects were found, were also used to provide them with the weight – from 1 point for the internal defects to 100 points for the ones reported from field or from the customers.

We have also found that dependency on other teams cannot be quantified given the current set-up and therefore the information need can be satisfied only through the expert assessment at this time.

The dashboard for the first information need is presented in Figure 2.

The figure shows how the stakeholders monitor the problem reports – both as individual numbers and as trends. Clicking on each of the widgets leads to the details of which problem reports are counted and their details.

5.2 Defects carried over into integration

Three roles were identified as stakeholders for the category of defects into integration: integrators (engineers who are responsible for integrating software components into subsystems and software with hardware), testers (responsible for the development, execution and monitoring of integration, system, regression and function tests), and the development team (DevTeam, responsible for the design and implementation of software requirements).

The information needs and the measures are provided in Table 4.

We have found that this category was very important for both the entire company, and for the particular stakeholders. The measures in this category monitor the quality of the software and the potential risk that the software requires re-work and thus increased costs. The number of elicited information needs was higher than in the previous category, becuase of the higher number of stakeholders.

We can also observe that, compared to the standard SAFe measures, these information needs and measures are related to product and organization/infrastructure status. The audience is more towards the technical side of the development organization, not the project or product management.

The dashboard which realizes these information needs is presented in Figure 3. The numbers with orange background correspond to the information needs realized:

- What is our test status? – widgets 1, 2 and 3,
- What is the status of our test rigs? – widgets 4, 5, and 6, and
- What is the availability of our tools? – widget 7.

Clicking on an area in the dashboard leads to details related to the measure, e.g. rig availability over time.

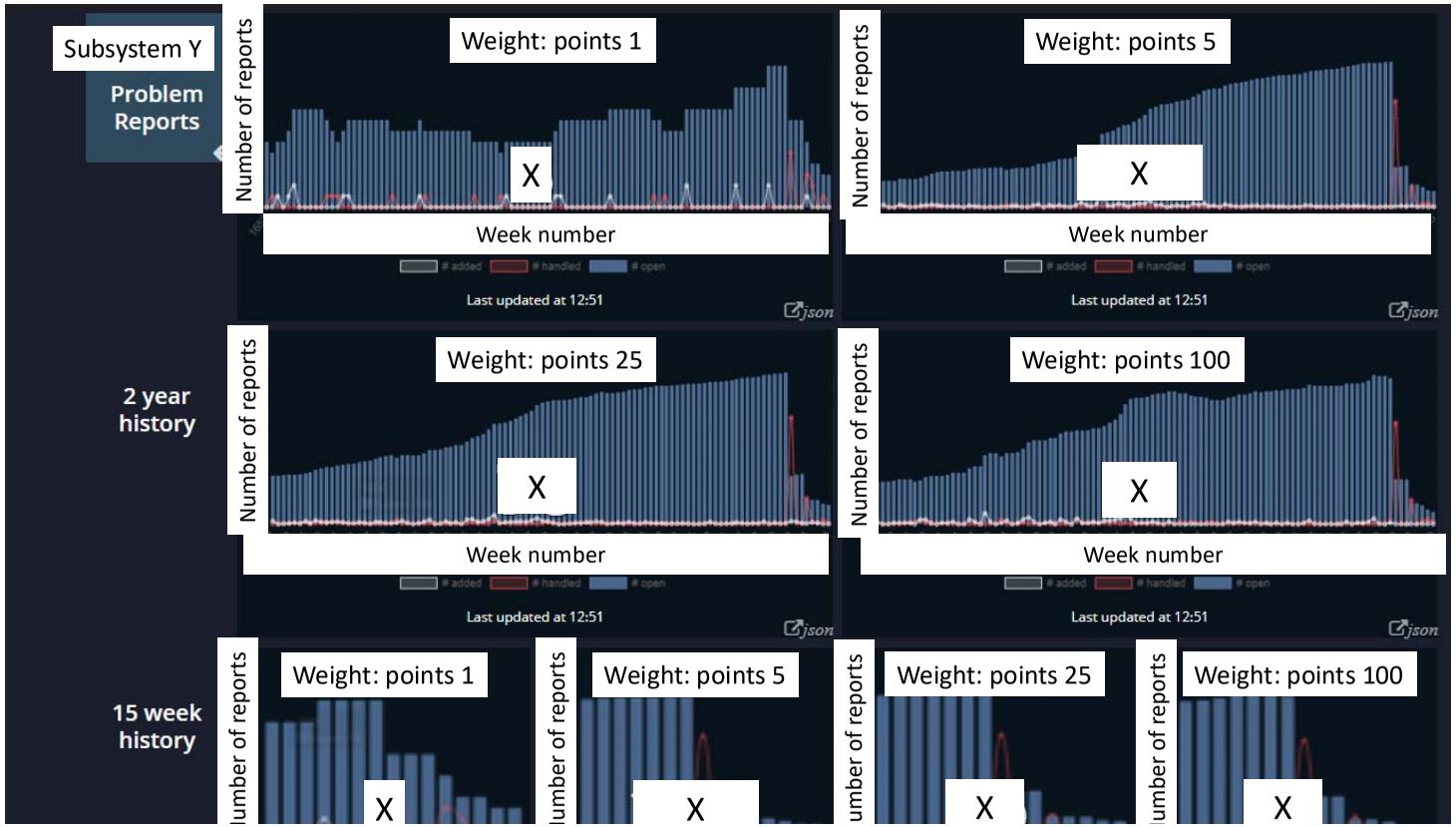

Fig. 2. Dashboard for problem reports from the field. The company specific information and the numbers have been covered due to non-disclosure agreements with the company.

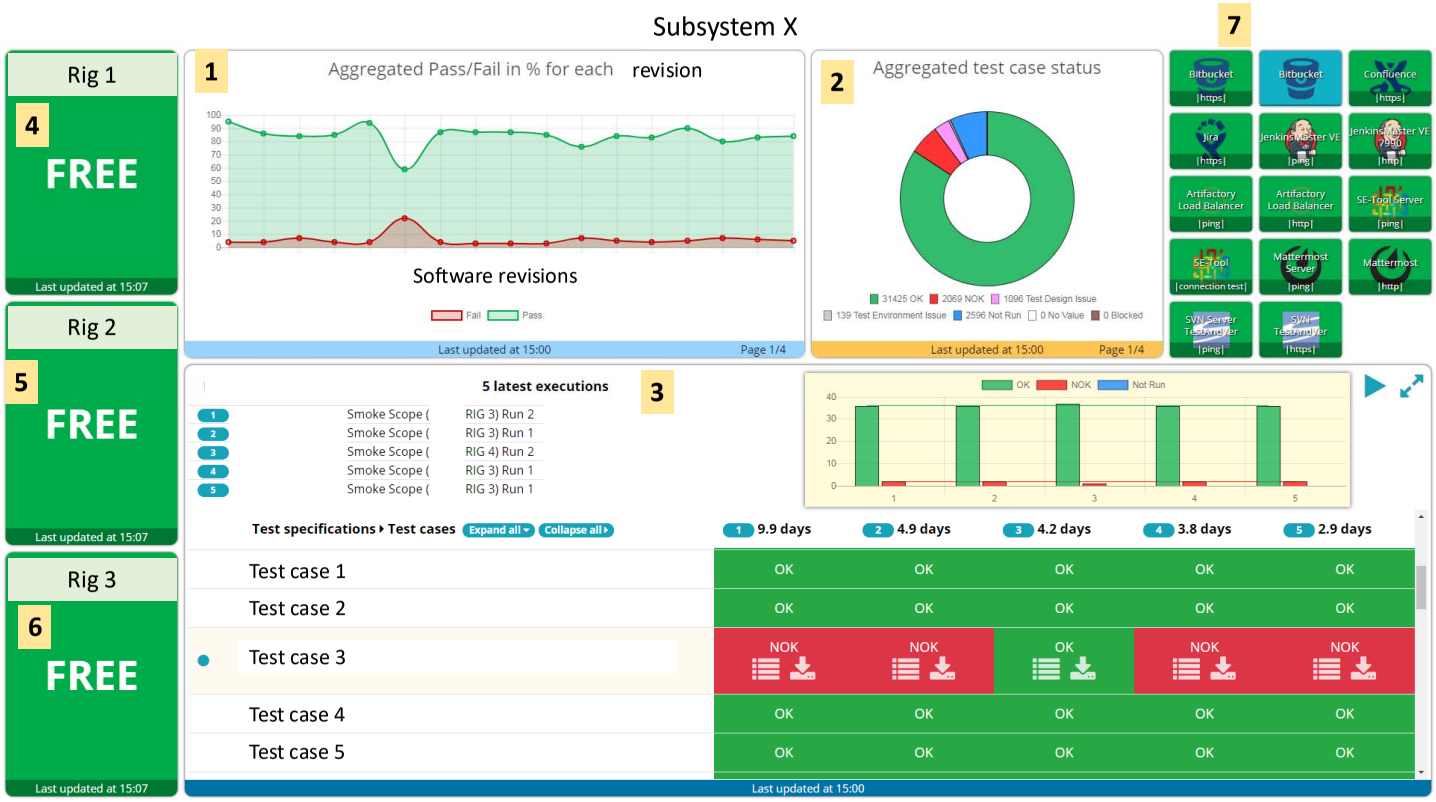

Fig. 3. Dashboard for defects into integration

Table 4. Measures for defects into integration

Stakeholder	**Information Need**	**Measure(s)**
Integrator	What is our integration status?	Days to integrate
Integrator	What are the versions of the used tools?	N/A
Integrator, DevTeam	What is our integration speed?	Build time
Integrator, DevTeam	What is our integration speed?	Compilation time
Integrator, DevTeam	What is our integration speed?	Generation time
Integrator, DevTeam	What is our integration speed?	Time to create the description file
Integrator	What is the size of our software?	Artifact size (binary file size)
Tester	What is the availability of our tools?	Binary status of tool availability
Tester	What is our test status?	Uptime / average time between fails
Tester	What is our test status?	Response time for server
Tester	What is our test status?	Test results over time (per sw. revision)
Tester	What is our test status?	Requirements coverage over time (per sw. revision)
Tester	What is our backlog?	Average number of sw. defects over time
Tester	What is the extra workload in our sprints?	Burn-up over time (work items not planned)
Tester	What is our planning accuracy?	Schedule slippage
Tester	What is our release speed?	Time between the designer's readiness of model and model's release
Tester	What is our test status?	Number of smoke tests executed
Tester	What is our test status?	Number of scope tests executed
DevTeam	What is the status of our test rigs?	Test rig availability
DevTeam	What is the quality of our integration?	Code commits/broken builds
DevTeam	What is the status of our release?	Status of release steps per sprint
DevTeam	What is our backlog?	Number of changed Electronic Control Units (to be integrated)
DevTeam	What is our integration speed?	Total lead time from model to integrated code
DevTeam	What is our test status?	Number of passed regression test cases
DevTeam	What is our test speed?	Execution time per test case
DevTeam	What is our defect resolution speed?	Defect resolution time

The dashboard provides an overview over the status of the defects carried over to the integration. It shows the status of the testing equipment, the test progress and the test status for the latest few tests.

The version of the tools used, which fulfills the second information need, is presented in Figure 4.

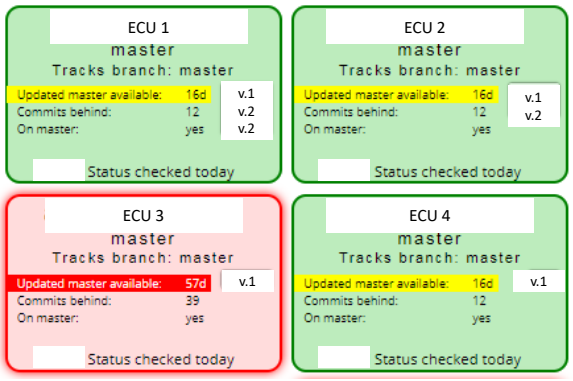

Fig. 4. Dashboard for versions of the tool used (only four Electronic Control Units shown in the diagram)

One of the control units has a software that has not been updated for a long time, and therefore the status for this control unit is red.

5.3 Integration status

The stakeholder for the integration status is the program manager. The results are presented in 5.

Table 5. Measures for integration status

Stakeholder	Information Need	Measure(s)
ProgMan	What is our integration and PI status?	Deviations (in days) of project plan dates from program increment dates
ProgMan	What is the speed of our integration?	Build throughput
ProgMan	What is the speed of our integration?	Build interval
ProgMan	What is the speed of our integration?	Build lead time
ProgMan	What is the capacity of our integration?	Rig status (available, not available)

We have found that the integration status is important from two perspectives – the availability of the integration toolset (which was already discussed in the previous category – defects into integration), and the speed. The information need of speed related to the management's need to understand whether the organization is performing with the optimal scope and optimal speed compared to the resources.

6 Validity evaluation

To discuss the validity of our study, we use the framework advocated by Wohlin [19].

The main threat to the *construct validity* of our study was the fact that we used workshops as the main data collection procedure. To minimize this threat, we developed prototypes which were used at the company (examples presented in the paper), to check whether the results are relevant.

One of the main threats to the *internal validity* is the fact that we did not use recordings or transcripts for the workshops. We chose this as the better option as it provided us with the more open environment. Our countermeasure to avoid the bias in notes was that two authors took notes independently.

In the *conclusion validity* category, we minimized the threats by conducting analyses independently by two authors. We collected the notes independently and one of the authors checked the consistency between these notes.

Our study has been conducted at one company only, which creates the threat to the *external validity* about the generalizability of the results. In order to minimize the threat to validity, we presented the results to other companies as part of our research project.

7 Conclusions

In this paper we explored the information needs of an organization developing, integrating, testing and deploying embedded, safety-critical software. We started by analyzing the existing measures prescribed by the process adopted by the organization – SAFe. We have found that the prescribed measures are not sufficient, they lack the focus on product and focus on the process.

Our design research resulted in the identification of 28 new information needs. These information needs were fulfilled by a similar number of measures (some information needs required expert assessment instead of measurement). These results are aligned with the previous work of our team [9], conducted at another organization. Both cases resulted in the complement of the prescribed measures with the domain specific ones.

In the further work, we envision the study of more organizations adopting similar processes and designing a portfolio of Quality Measure Elements (according to the templates of the ISO/IEC 25000 standards). These measures could be used as prescribed in the SAFe processes.

References

1. Augustine, V., Hudepohl, J., Marcinczak, P., Snipes, W.: Deploying software team analytics in a multinational organization. IEEE Software (1), 72–76 (2018)
2. Bosch, J.: Continuous Software Engineering. Springer (2014)
3. Bosch, J.: Speed, data, and ecosystems: The future of software engineering. IEEE Software **33**(1), 82–88 (2016)
4. Davis, C.W.: Agile Metrics in Action. Manning Publications, (2015)
5. Huijgens, H., Lamping, R., Stevens, D., Rothengatter, H., Gousios, G., Romano, D.: Strong agile metrics: mining log data to determine predictive power of software metrics for continuous delivery teams. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, pp. 866–871. ACM (2017)
6. ISO, I.: 26262-1: 2011 (en) road vehicles-functional safety-part 6: Product development at the software level, 2011
7. Leffingwell, D., Yakyma, A., Jemilo, D., Knaster, R.: Scaled agile framework. Siehe: http://scaledagileframework. com (2013)
8. Liechti, O., Pasquier, J., Reis, R.: Beyond dashboards: on the many facets of metrics and feedback in agile organizations. In: Proceedings of the 10th International Workshop on Cooperative and Human Aspects of Software Engineering, pp. 16–22. IEEE Press (2017)
9. Meding, W.: Effective monitoring of progress of agile software development teams in modern software companies: An industrial case study. In: Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement, IWSM Mensura '17, pp. 23–32. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3143434.3143449. URL http://doi.acm.org/10.1145/3143434.3143449
10. Organization, I.S., Commission, I.E.: Software and systems engineering, software measurement process. Tech. rep., ISO/IEC (2007)
11. Ostakhov, V., Artykulna, N., Morozov, V.: Models of it projects kpis and metrics. In: 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), pp. 50–55. IEEE (2018)
12. Schermann, G., Cito, J., Leitner, P., Gall, H.C.: Towards quality gates in continuous delivery and deployment. In: Program Comprehension (ICPC), 2016 IEEE 24th International Conference on, pp. 1–4. IEEE (2016)
13. Staron, M.: Automotive Software Architectures: An Introduction. Springer (2017)
14. Staron, M., Meding, W.: Software Development Measurement Programs: Development, Management and Evolution. Springer (2018)
15. Staron, M., Meding, W., Hansson, J., Höglund, C., Niesel, K., Bergmann, V.: Dashboards for continuous monitoring of quality for software product under development. In: Relating System Quality and Software Architecture, pp. 209–229. Elsevier (2015)
16. Staron, M., Meding, W., Karlsson, G., Nilsson, C.: Developing measurement systems: an industrial case study. Journal of Software Maintenance and Evolution: Research and Practice pp. n/a–n/a (2010)
17. Staron, M., Meding, W., Nilsson, C.: A framework for developing measurement systems and its industrial evaluation. Information and Software Technology **51**(4), 721–737 (2008)
18. Wieringa, R.J.: Design science methodology for information systems and software engineering. Springer (2014)

19. Wohlin, C., Runeson, P., Host, M., Ohlsson, M.C., Regnell, B., Wesslèn, A.: Experimentation in Software Engineering: An Introduction. Kluwer Academic Publisher, Boston MA (2000)