Building an Effective Software Issues Scorecard: An Action Research Report from the Automotive Domain

Rakesh Rana
Skövde Artificial Intelligence Lab
University of Skövde
Email: rakesh.rana@his.se

Tommy Lagercrantz
Volvo Car Group
Göteborg, Sweden

Miroslaw Staron
Chalmers - University of Gothenburg
Göteborg, Sweden

***Abstract*—A large number of mature software companies use data and analytic for status monitoring of their projects and to help improve their decision making at different levels within the organization. Dashboards or scorecards also provide common platform for different stakeholders to access information they need for tracking the status of projects of their interest. Further data from software issues database can provide real and observable indicators to track the quality of given product during its development and testing. The study presented here reports on distinct and evolution of information needs of different stakeholder groups interested in tracking such data. The action research report documents the evolution of software issues scorecard as it is extended to meet information need of specific user groups. A roadmap for future into how such scorecard can be made more effective is also presented.**

I. Introduction

Finding and fixing defects (or issues) is overall the most expensive activity in embedded software development [1]. Given the size, complexity, time and cost pressures - tracking and predicting quality is a major challenge in automotive software development projects. To meet the demands of high quality and reliability - significant effort is devoted on software V&V (Verification& Validation). Testing the software is an important part of software V&V used for ensuring correct functionality and reliability of software systems; but at the same time software testing is also a resource intensive activity accounting for up to 50% of total software development costs [2] and even more for safety critical software systems. Thus having a good testing strategy is critical for any industry with high software development costs.

One way of being efficient in dealing with software issues is an efficient way of organizing discovered defect reports to facilitate effective exchange of information between testers and developers, constant monitoring of found issues until their resolution and validation. Also by utilizing better visualization techniques for data representations, the software issues data can be made more useful and interactive for its stakeholders that allow for fast response. Primarily software issues and reliability measures are used for [3] [4]:

- Software process improvement,
- Planning and controlling testing resources during software development, and
- Evaluating the maturity or release readiness of software before the release date.

Most organizations maintain defect databases which can be local to a team, project/product or specific section of an organization. All defects found during verification and validation activities are reports in these databases in a predefined format - often with the sole purpose of facilitating their resolution. The database usually provides the platform where different stakeholders within and outside of an organization can:

- Access the information about issue(s) of their interest,
- Add, edit, or update the information related to a given issue,
- Comment, provide expertise or guidance to help resolve the issue, and
- Track the progress of reported issue(s) and monitor defect statistics.

To facilitate the documentation and exchange of information, various attributes are recorded for each reported software issues. Some of these attributes are mandatory aimed at providing the basic information pertaining to given defect, while others are optional that provide additional details. The overall goal is to provide information from actor (usually tester) who discovered the defect to actor(s) who will resolve or help resolve it (usually developers).

The main challenge is of course how to make the information documented and available in these defect databases interactive and user friendly for different stakeholders. In this action research we specifically studied the information needs of different stakeholders, and how the Software Issues (SWI) scorecard could cater to their information needs.

The rest of paper is organized as follows, the next section following this introduction gives a brief overview of related work. In section III we discuss the research methodology and providing more details about the case company and its software development and testing process. Section IV provides the overview of state of SWI scorecard at the beginning of

the study, followed by section V documenting the evolution of same as the study progressed. The study is summed with conclusions presented in section VI.

II. Related Work

Large software projects within the automotive domain particularly the development of platform projects at full EE (Electronics & Electrical System) level can span over several months and include number of iterations/integration points [5]. The combination of large size of projects, rapid feature development, and extensible architecture implies emergence of new information needs [6] across projects and also during the project evolution. Assessing the information need of different stakeholders and meeting those needs using data from different sources is an important activity to ensure the development of high quality software within the planned resources and time.

Dashboards have been used to monitor number of important metrics or KPIs (Key Performance Indicators) evolution over time [7] [8] [9]. Richard W. Selby [10] contend that measurement-driven dashboards provide a common platform for understanding, evaluating, and predicting the development, and management of large-scale systems and processes. The author provided empirical observations of dashboards that have been used on actual large-scale projects. Buse and Zimmermann [11] propose using software analytics for helping mangers to meeting their information needs. Baysal, Holmes, and Godfrey [12] motivates the need for qualitative dashboards specifically designed for the need of developers to help them improve their situational awareness by providing custom views of variety of information related to their tasks that will help them manage their workload while performing day-to-day development tasks.

Visualizing large number of measure in one place helps upper management and teams [13]; it can help upper management to monitor and control development process and self-organized software development teams can use it to monitor and communicate the status of the quality of their product under development [14]. The basic requirements for such dashboards are that they need to collect important and necessary information and visualize it in a simple way which is intuitive for its intended audience. Staron et al. in a book chapter [15] used case study from three large companies engaged in software development provides an overview of develop and use dashboard for monitoring of software development progress and discuss the quality of software architectures under development. In this paper we describe the scorecard for software issues used within a large automotive OEM (Original Equipment Manufacturers), the main purpose of this scorecard is to collect and summarize the information for software defect database for a given ongoing project to help software development and test teams, sub-system (ECU) owners, quality and project managers on get the latest updated information (with respect to software issues) in an user friendly way that is easy to interpret by both technical and not so technical stakeholders.

Another characteristic of most dashboards used in industry are that they evolve over time, as the use of dashboard is increased among its intended (and sometimes extended) audiences new information needs are communicated by different stakeholders. These information needs may be due to specific interest of particular group of stakeholders or due to evolution of software development/testing/management process itself which renders monitoring of certain new information/metrics important. Rotella and Chulani [16] reported the implementation and evolution of software quality goals (and corresponding Software Quality Dashboard, SWQD) at Cisco Software. The authors also reported number of quality improvements achieving the quality goals during the studied period. This paper also reports the evolution of software issues scorecard during the studied period providing details on the information needs of different stakeholder groups and how they were fulfilled.

III. Research Methodology

Given the main objective of this research study was to understand the evolving information needs of current users of SWI scorecard by different stakeholders in the context of large automotive software platform projects and help bring in changes to improve its effectiveness as information tool, action research methodology was used in this research study. Action research is a research strategy for organizational research in which the researchers are themselves involved in the studied change. In action research, new knowledge is created through seeking of solutions or improvements to real-life practical problems [17]. The general process cycle for action research utilized in this study can be presented as Figure 1.

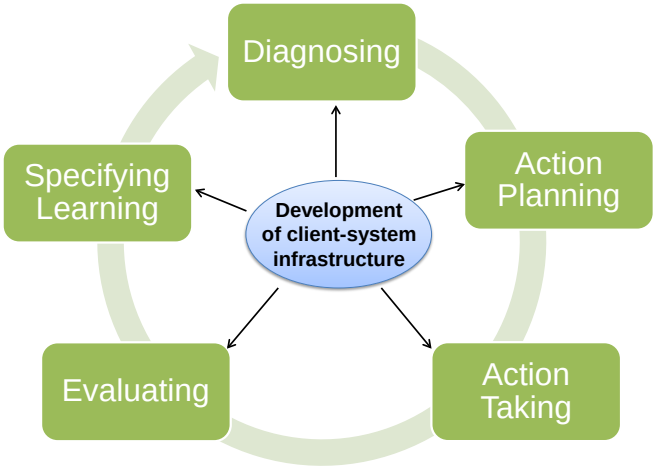

Fig. 1. Overview of action research cycle used in the study

The primary actors involved in the action research were,

1) **Metrics Team Leader:** The Team leader responsible for collection, analysis and reporting of project status with regard to software issues to different software teams and management within the company, the team leader has more than three decades of experience in various roles at the company. Researcher: The researcher involved in the reported action research project has worked with the

case company for over three years on various research projects.

2) **Stakeholders (scorecard users):** The stakeholders that have contributed to this research study mainly include regular users of SWI scorecard which constituted team leaders of various software development and test teams mainly at sub-system/ECU level, system/function owners, project, and quality managers.

As expected in a study of this nature and also according to the illustration of Figure 1, the research study was conducted in iterative manner with inputs from one group of stakeholders collected, analysed and if deemed important implementation was done to meet the requested information need. Following the implementation, enough time was given for all regular user to adapt to new changes and the updates were followed up (for evaluation and learning) with the user group that requested the implemented changes but also with other user group on their assessment of added functionality.

A. The Case Company: Volvo Car Group

Volvo Car Group (VCG) is a Swedish car Original Equipment Manufacturer (OEM), based in Gothenburg. VCG develops software and hardware typically in a distributed software development environment, but for a limited number of Electronic Control Units (ECUs) the software is also developed in-house. The development is done by the software development teams who usually also hold responsibility for integrating the software with the hardware developed by suppliers. The majority of the embedded software development in the car, however, is developed by external suppliers who design, implement, and test the functionality based on specifications from VCG ([18], [19]). SWI scorecards we worked with during this study are the large platform projects, the project come from the EE (Electrical and Electronics) integration department within the VCG which deals with the integration of various software functionalities and responsible for the final assessment of full EE hardware and software systems.

B. The SW Development Life Cycle & Testing Process

Most automotive Original Equipment Manufacturers (OEMs) follow Model Driven Development (MDD) and since car platform projects are often large and spread over several months, they are executed in number of iterations. In literature and development standards, software development life cycle in automotive domain has been illustrated as approaches based on V-model [20], [21]. The life cycle of full EE car platform projects have been described in authors earlier work [5], in simple terms it can be represented as shown in Figure 2.

The process followed at each iteration within the production software development phase can be described using a V-model (refer to Figure 3), essentially for each iteration - first the requirements are set or reviewed followed by System Design (functional design and system architecture). Following the system design ECU specifications is done which can also be referred as software design since software is usually designed for specific ECUs and they are generally co-developed, optimized for particular functionality.

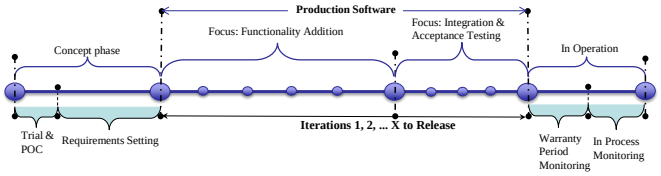

Fig. 2. Representation of software development life cycle for EE platform project within automotive domain

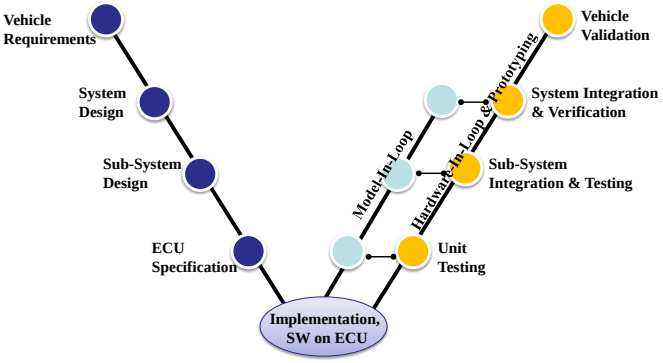

Fig. 3. Overview of software development process at VCG

Next comes the implementation where designed software is implemented (as code either manually written in object oriented language or auto-generated from a functional model build using some domain specific language (DSL) such as Matlab/Simulink). The implemented code usually undergoes rigorous testing under simulated environment to ensure correct working of intended functionality and fulfilment of desired quality requirements. The testing of software in simulated environment is termed Model-In-Loop testing where different functional models/code is also integrated and tested. The software code is then integrated within the hardware/ECU and it follows the testing via the Hardware-In-Loop testing (for all iterations) and testing within complete vehicle prototypes (for certain iterations). Major types of testing carried out to verify and validate the functionality include unit testing, sub-system integration and testing, system integration and testing, functional and acceptance testing.

IV. SWI Scorecard

The main idea behind using a SWI scorecard is to provide a snap shot view of current status with respect to software defects for a given project. The SWI scorecard is basically an outcome of applying balanced scorecard [22] approach for reporting the Key performance Indicators (KPIs) specifically to software defects discovered and resolved in the given project at any given point in time. Given that SWI scorecard provides KPIs and summary statistics related to SW defects of a given project, it is primarily aimed at team leaders, system owners, quality, and project managers to keep track of the progress

of project from software defects perspective (an important indicator of software quality and maturity). With the objectives and target audience in mind, the two basic features required from a good SWI scorecard would be:

- It provides a good overview of current status of project (with respect to SW defects),
- It is updated frequently to ensure the latest data is available, and
- The information is provided in format that is easy for its intended users.

A. State Of SWI Scorecard at The Beginning of Study

At the start of the action research reported here, the case company has SWI scorecard that satisfied the basic requirements outlined above. Basically the scorecard provided overview of information deemed most important by various stakeholders and also included graphical representation for easy interpretation. The basic information conveyed included,

- Number of SW issues/defects found until the given time,
- Status (open, closed, etc.) of SW issues found until the given time, and
- The planned follow up of all unresolved issues (planned fix date, test and validation plan).

The information was presented both in tabular and graphical format that made it fast and easy to interpret for different stakeholders of the scorecard. The scorecard was built on a popular spreadsheet application linked to the software issues database, the updated scorecard sent out to relevant stakeholders (as well as available in a shared space). A simplified version of how the information was provided graphically is shown in Figure 4.

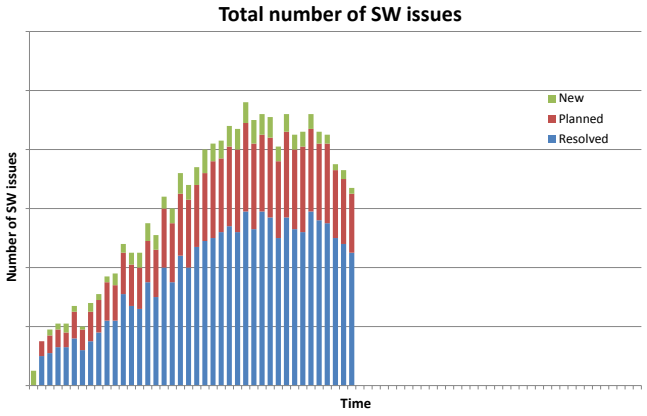

Fig. 4. A simplified representation of graphical data visualization in original scorecard

The figure shown here (Figure 4) is only a simplified representation for illustrative purpose, the actual view cannot be included due to confidentiality, but the representation illustrates well the basic status representation in the original graphical presentation. The figure provides its users a snap shot of most desired information with respect to issues/defect statistics for a given project at a given point in time and its historical evolution. It also provide the statistics with regard to future plans (i.e. the test and validation plans).

B. Understanding the Information Needs

While the SWI scorecard meet the basic expectations of its intended users, the primary participant of action study (the metrics team leader and researcher) wanted to find out potential areas of improvement with the aim of making the SWI scorecard more user friendly and one that meets more than basic needs of its users.

To better understand the needs of SWI scorecard users, the metrics team leader and researcher collected views from sample of it users. The views of users were collected during regular meeting metrics team leader attended with different teams, unstructured and semi-structured interviews with different groups of scorecard users. Attempt was made to collect suggestions and opinions from representatives of each group which included:

- Team leaders of software development and testing teams,
- Sub-system/ECU owners (that included number of function teams),
- Function owners (logical functions/features can have components across multiple ECUs), and
- Quality and project managers at different levels.

A number of suggestions for improvement were collected during the informal talks with the stakeholders, specifically the opinions that came across frequently were:

- The need to drill down from the project to sub-system level, this was mainly requested by team leaders and ECU owners so that they can track their teams progress and also compare it to other teams working on the same project,
- It was expressed that the most recent data (for e.g. data from last x weeks) could be presented separately, since this was most interesting at the given point in time,
- Function owners wished for ability to track for a specific combinations of teams at a time, and
- Quality and project managers expressed their desire to see some more aspects such as defect discovery rate, defect resolution rate and where possible some projections in future.

The different views and suggestions offered by different groups of stakeholders were deemed useful and thus SWI scorecard was evolved to accommodate these suggestions. The next section provides more details on how the scorecard evolved following updates to meet the new requirements.

V. SWI Scorecard Evolution

In this section we describe the features or new information integrated to the SWI scorecard which resulted in its evolution during the action research. As described in previous section, the information need was first assessed by talking to relevant stakeholders and users of SWI scorecard. Each informational element was added one at a time and given time to sink in with its users. Keeping changes one at a time and maintaining

a set period before next change were designed to allow users to adopt to new information in the scorecard, and provide feedback with regard to its suitability. The metrics team leader made sure that all changes in the SWI scorecard were introduced and explained to a sample of its uses in different meetings and forums. A user manual was also sent out with the SWI scorecard when the change was large and required explanation on how to use it.

Given that the original SWI scorecard have been developed as an excel template, wide familiarity of Microsoft Excel within company and the fact that new functionality in latest version of Excel allowed extending the scorecard as desired contributed in keeping excel as the scorecard reporting tool.

A. Filter By Functions & ECUs

The first information that was added to original SWI scorecard was the ability to easily filter SW defects statistics based on individual functions and ECUs. Since many of the scorecard users were either team leaders responsible for a specific ECU or function owners, the filtering ability at the granularity of functions and ECUs was very well received. The filters were added using excel built-in pivot tables and slicer function that allowed users to:

- Filter data on individual functions/ECUs,
- Filter data on specific combinations of functions/ECUs this functionality was useful for managers responsible for several teams, and
- View the charts for filtered data (the display charts get dynamically updated based on the chosen filter)

An example of filter slicer is shown in Figure 5, a user can chose a desired individual (or combination) of functions/ECUs simply by clicking (CTRL+click) on functions/ECUs of interest. The user is able to go back to project level (default view) data by cancelling the filter using cancel filter tab on upper right corner of slicer.

This feature not only proved useful for individual team leaders and function owners for assessment of their team SWI statistics, but it was found out that it found much use during cross-functional team meetings where it provided a snap shot view of current status of different teams with respect to SW issues and encouraged healthy discussions and exchange of information between cross functional teams.

B. Discovery & Resolution Rates

Another important statistics with respect to SW defects, different stakeholders were interested in was the rate of discover and resolution at given point in time and how it has been since the start of project. Although it was possible to infer these rates from the main chart in original SWI scorecard - it was not very intuitive/user friendly. The main chart in original scorecard basically showed historical data on number of defects found and resolved against time in bar chart - which meant that to infer their rates users had to follow the trend of bar chart over time.

To resolve the aforementioned difficulty, firstly two line plots were added to the main chart showing the rate of defects

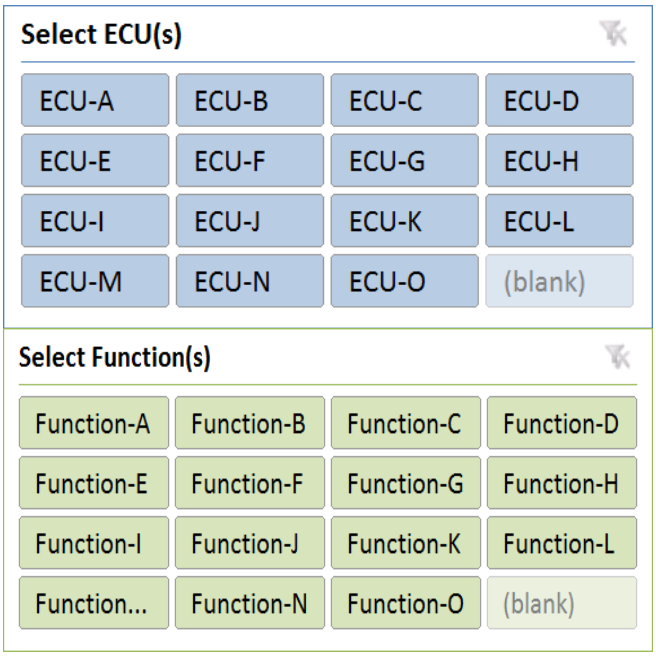

Fig. 5. The representation of slicer function to allow users to filter information on specific systems/sub-systems

found and resolved now users can see the evolution of two rates (defect discovery and resolution) in a quick glance and the gap between them also provided simple measure of if the verification and validation activities were on track. In order to give users also an easy way to check the numbers associated with these rates, a new chart was added to the new scorecard with simple bar chart that showed number of defects found and resolved in given week side by side. Simple illustrations of inclusion of discovery and resolution rates on main chart and additional bar chart are shown in Figure 6.

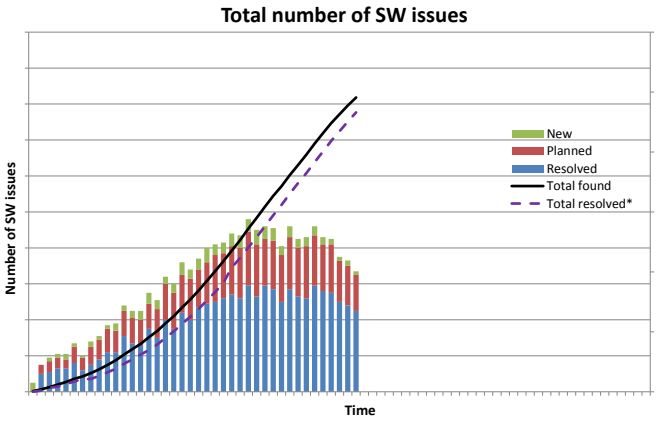

Fig. 6. Representation of discovery and resolution rates to one of original scorecard graph

As it can be seen from the figure, the additions of lines in main char and the added bar chart provides the desired information in an intuitive manner. It is easy to identify the

time periods when the rate of defect discovery &/or resolution was fast/slow and how many defects were found and resolved at a given point in time.

C. Last Time Period, Shorted

Quality and project managers responsible for full project or large number of teams usually have to shift through large amount of information from widely different spectrum related to given projects. They also use SWI scorecard for monitoring of project and teams with respect to software defects. One aspect these managers are interested in is quickly identifying specific teams with either largest reported issues or ones that may have deviated from their verification & validation plan. This is mainly to identify teams where these managers need to focus at given point in time and using SWI statistics provide on source of information to making such decisions.

To cater to these needs a new filter was added to the scorecard that can short functions/ECUs on the most important SWI attributes such as number of defects found, number of defects resolved, difference between found and resolved, average and median age of unresolved software issues, deviations from validation plan etc. Another important feature of this added filter was ability to select the time period of interest, which is usually last couple of weeks for most mangers. Once the attribute of interest and time period is selected by a manger, the scorecard lists the individual functions/ECUs with values shorted such that teams that may need some attention are listed on top, while teams meeting the expectations are listed below. This allows managers to combine this information with other sources of information to decide if they wish to follow up with a particular team and dig deeper on any aspect or not. A basic representation of filter is shown in Figure 7.

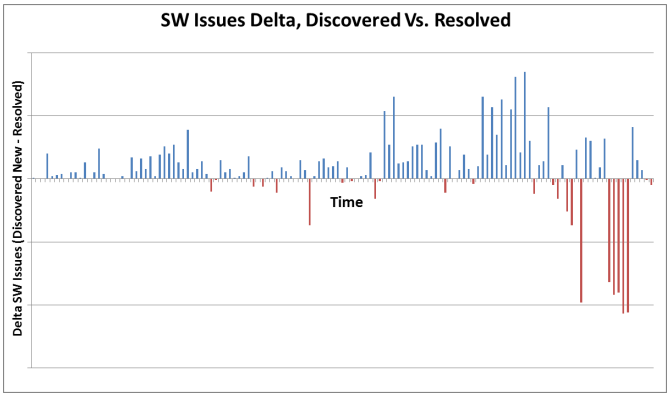

Fig. 7. An example of added graphical representation of information added to original scorecard to show delta issues

D. Other Attributes of Interest

The defect databases record number of different attributes. Although the primary aim of such documentation is to facilitate the information exchange between person who discovered (usually testers) and reported the defect into the database and actors that will resolve it or help in its resolution; the information available in these database can be used to get more insights into the software development and testing process and thus help in software process improvement.

Software defect classification schemes can be can be used to develop templates for defect reporting that share a well-defined structure. Such pre-defined and shared structure facilitates quantitative analysis of defect reports that can provide useful insights to characterize the development process and also assist in identifying improvement opportunities [23]. Examples of defect classification schemes include orthogonal defect classification [24] developed at IBM, schemes based on IEEE standard classification for software anomalies (IEEE Std. 1044) and a light-weight defect classification scheme [23].

In this sub-section we list using one such attribute from the defect reports that we use for further insights, but given the information need and desire to explore different aspects of software process within a given company different attributes recorded in the defect databases can be leveraged for more in-depth analysis. Since the large E/E platform projects within automotive domain involve large number of ECUs, integration testing is an important part of each iteration/integration point. Before integration points, each team tests their software for unit and function tests and following integration, integration, function and acceptance testing is usually done by various independent teams with our studied company. We wanted to find out more on which teams were contributing to defect discovery to the software of one particular ECU that have many functions communicating (dependencies) with several other ECUs. It was also intended to see how the contributions evolved over time as the platform project progressed. Figure 8 show the share of different teams in number of reports software issues over first four major integration points.

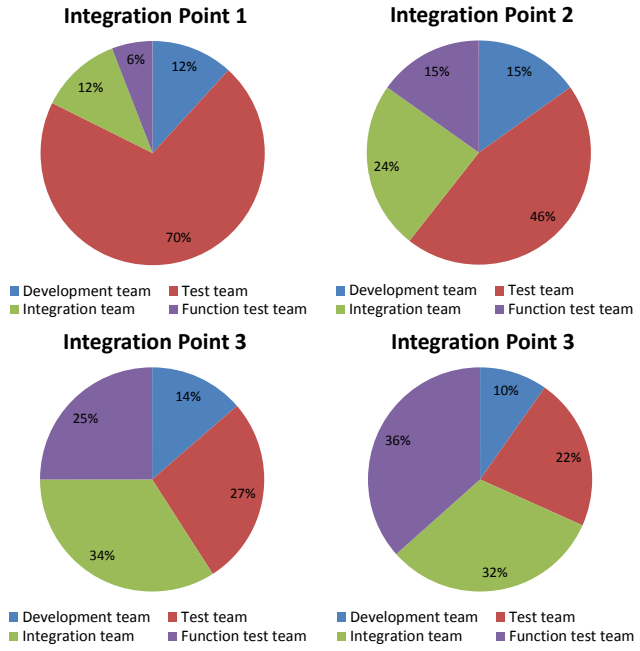

Fig. 8. Representation of using graphical representation of information for deeper insights from SWI issues data

The pie charts display very clearly and provide a simple overview of how much of software issues were reported (proxy for amount of testing) by:

- Software development team for this ECU,
- Software test team for given ECU,
- The independent integration teams at integration department, and
- The team responsible for verification and validation at function level.

Following this further into late integration points, one would be able to see the share of software issues reported by function testing, and teams conducting acceptance test at vehicle level. Such deep analysis of specific attribute(s) from the defect database can help provide empirical evidence for or against common held beliefs, evidence to test certain hypothesis about software development or test processes and provide insights that can be used for process improvements.

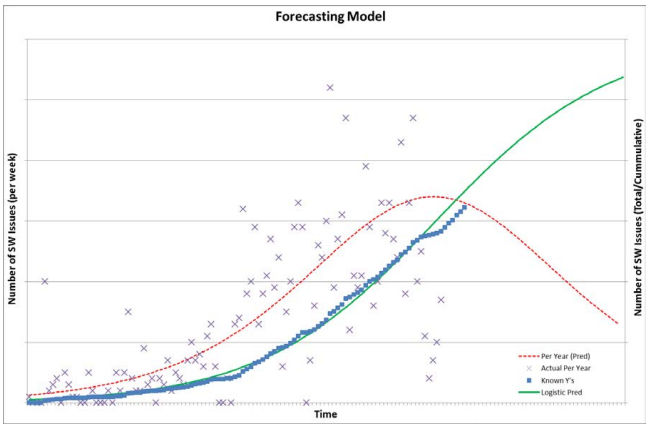

Fig. 9. Representation of forecasting model added to original SWI issues scorecard to fulfil trend projection needs

E. Projections or Forecasting

Another feature mainly requested by quality and project managers was the ability to see some projections into future. Since the defect discovery and resolution rates are usually not linear for e.g. the widely reported S-shaped defect inflow theorizes that the defect discovery rate is low at the beginning of testing, maximizes around middle and again slow down toward the end of testing as the software under test matures with resolution of discovered defects.

Quality and project managers who are responsible for delivering high quality software in a given time frame are often interested in knowing if the software under development and testing will reach the desired quality/maturity demands by the given release date. Thus projections using current defect inflow data and based on theoretical software reliability models can provide useful projections into future to help quality and project managers' asses if the project under development or testing is on track or would require more resources.

We added visual basic routine in Excel SWI scorecard template, which can use the partial defect data for given project and fit logistic or gompertz model on the data to provide future forecasts. For details on use of software reliability growth models for optimal test resource allocation and release readiness assessment in the automotive domain, readers are referred to authors earlier work [25]. The Figure 9 show an example of using logistic model to fit the observed defect data and providing projections based on the logistic growth model.

It is noted here that the implementation of this routine is an recent effort and requires some manual inputs, but still it does provide important advantages for industrial practitioners in terms of shielding them from the need to manually filter the data from defect database, exporting it to some statistical software package and using complex mathematical models to obtain projections using software reliability models. Once enough testing (defect) data is available, about halfway through the project [26], the routine is able to provide stable forecasts that can help managers monitor and asses the current status of projects from the software defect management perspective.

F. Roadmap for Future

During the course of this action research, the SWI scorecard evolved from a simple excel template providing basic defect statistics information to a platform with rich information set to address the specific information needs of different user groups. Nonetheless there is much more that can be done to continue the process of building an effective software issues scorecard. The most notable future work is listed here:

- **Online:** The choice of continuing with Excel as the basic reporting tool provided many advantages, specifically familiarity among user groups, easy display of data in form of tables and charts. But it also mean that the scorecard needs to be updated manually and available as the latest version document (offline). It is recognized that within near future, the SWI scorecard need to be made online such that anyone can access the scorecard using standard web browser or even mobile application and the scorecard would be updated automatically (data synchronized with the defect database).
- **Different views:** While the new SWI scorecard provides important information that meets demands of different user groups, currently all this information in form of tables and charts are presented in the same file. This have two important limitations in terms of user friendliness, the first that the file becomes large over time and secondly and more importantly that the views are not customized for different user groups. Although temporary solution is made available by using different sheets to address different user groups, but over time as the scorecard is made available online different views would be implemented such that the view and presented information would be customized to the particular group of users.
- **More visualization capabilities:** One of the main outcome of presented action research has been recurring

requests and implementation of visual representation of data for quick and intuitive understanding. Nonetheless it is still recognized in the project that more visualization capabilities can be added to current state of SWI scorecard. Specifically by going online and implementing separate views for different user groups will allow for more room to add new data visualization. The need to move more towards interactive visualization has also been marked for the future updates of the scorecard that will allow users not only to get statistics and visualize data in static form but more capabilities to interact with visual representation of data for digging deeper in areas of interest.

VI. Conclusions

Software issues provide a rich source of data that can be used by software development, test teams, and management to track and monitor the progress of on-going projects with regard to issues discovery and resolution. Since finding and fixing software issues represent a large part of software verification and validation activities and is a resource intensive activity monitoring such data is of high importance. The data if presented in an intuitive manner to different user groups can provide insights into the process and allow for tacking as well as software process improvements within organizations.

In this study we set out to understand the evolving information needs of different user groups of software issues scorecard in a large organization. In an action research setting we evaluated the needs of different stakeholders and document the evolution of SWI scorecard to meet these requirements. Given the nature of the objective of the research project, the study was set as an action research.

Over the study, different information needs from different stakeholders of SWI scorecard were collected, examined and where appropriate implemented to the evolving scorecard. A number of changes/features were added as a result which are documented in this action research report showing the evolution of such a dashboard (or scorecard). The general insights from the study include following observations:

- Different groups of stakeholders seek different information from software issues databases,
- The information need of different user groups usually evolve over time,
- The data form issues databases can provide useful insights into software process, and
- There is high need for better data visualization and interactive visualization.

Similar to most software systems and processes, most information dashboards or scorecards that are actually used in industry are living systems/documents. Documentation and more importantly evaluation of such evolution provides insights into the usefulness of such systems. It further helps to develop effective strategy and roadmap onto how such systems can be evolved to meet their main objectives and thus help companies to meet their organizational goals.

Acknowledgment

The work presented here has been funded by Vinnova and Volvo Cars jointly under the FFI programme (VISEE, Project No: DIARIENR: 2011-04438).

References

[1] C. Ebert and C. Jones, "Embedded software: Facts, figures, and future," *Computer*, vol. 42, no. 4, p. 4252, 2009.

[2] E. L. Jones, "Integrating testing into the curriculumarsenic in small doses," in *ACM SIGCSE Bulletin*, vol. 33, pp. 337–341.

[3] R. B. Grady, "Software failure analysis for high-return process improvement decisions," *Hewlett Packard Journal*, vol. 47, pp. 15–24, 1996.

[4] C.-Y. Huang, M. R. Lyu, and S.-Y. Kuo, "A unified scheme of some nonhomogenous poisson process models for software reliability estimation," *Software Engineering, IEEE Transactions on*, vol. 29, no. 3, pp. 261–269, 2003.

[5] R. Rana, M. Staron, J. Hansson, and M. Nilsson, "Defect prediction over software life cycle in automotive domain - state of the art and road map for future," in *ICSOFT-EA 2014 - Proceedings of the 9th International Conference on Software Engineering and Applications, Vienna, Austria, 29-31 August, 2014*, 2014, pp. 377–382. [Online]. Available: http://dx.doi.org/10.5220/0005099203770382

[6] R. P. Buse and T. Zimmermann, "Information needs for software development analytics," in *Proceedings of the 34th international conference on software engineering*. IEEE Press, 2012, pp. 987–996.

[7] S. Bouktif, G. Antoniol, E. Merlo, and M. Neteler, "A feedback based quality assessment to support open source software evolution: the grass case study," in *Software Maintenance, 2006. ICSM'06. 22nd IEEE International Conference on*. IEEE, 2006, pp. 155–165.

[8] N. Brown, Y. Cai, Y. Guo, R. Kazman, M. Kim, P. Kruchten, E. Lim, A. MacCormack, R. Nord, I. Ozkaya *et al.*, "Managing technical debt in software-reliant systems," in *Proceedings of the FSE/SDP workshop on Future of software engineering research*. ACM, 2010, pp. 47–52.

[9] J. Pérez, R. Deshayes, M. Goeminne, and T. Mens, "Seconda: Software ecosystem analysis dashboard," in *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*. IEEE, 2012, pp. 527–530.

[10] R. W. Selby, "Measurement-driven dashboards enable leading indicators for requirements and design of large-scale systems," in *Software Metrics, 2005. 11th IEEE International Symposium*. IEEE, 2005, pp. 10–pp.

[11] R. P. Buse and T. Zimmermann, "Analytics for software development," in *Proceedings of the FSE/SDP workshop on Future of software engineering research*. ACM, 2010, pp. 77–80.

[12] O. Baysal, R. Holmes, and M. W. Godfrey, "Developer dashboards: The need for qualitative analytics," *Software, IEEE*, vol. 30, no. 4, pp. 46–52, 2013.

[13] M. Staron, W. Meding, and C. Nilsson, "A framework for developing measurement systems and its industrial evaluation," *Information and Software Technology*, vol. 51, no. 4, pp. 721–737, 2009.

[14] H. Sharp, N. Baddoo, S. Beecham, T. Hall, and H. Robinson, "Models of motivation in software engineering," *Information and Software Technology*, vol. 51, no. 1, pp. 219–233, 2009.

[15] I. Mistrík, R. Bahsoon, P. Eeles, R. Roshandel, and M. Stal, *Relating System Quality and Software Architecture*. Morgan Kaufmann, 2014.

[16] P. Rotella and S. Chulani, "Implementing quality metrics and goals at the corporate level," in *Proceedings of the 8th Working Conference on Mining Software Repositories*. ACM, 2011, pp. 113–122.

[17] M. Elden and R. F. Chisholm, "Emerging varieties of action research: Introduction to the special issue," *Human relations*, vol. 46, no. 2, pp. 121–142, 1993.

[18] U. Eklund, N. Jonsson, J. Bosch, and A. Eriksson, "A reference architecture template for software-intensive embedded systems," in *Proceedings of the WICSA/ECSA 2012 Companion Volume*. ACM, 2012, pp. 104–111.

[19] R. A. McGee, U. Eklund, and M. Lundin, "Stakeholder identification and quality attribute prioritization for a global vehicle control system," in *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*. ACM, 2010, pp. 43–48.

[20] W. Dieterle, "Mechatronic systems: Automotive applications and modern design methodologies," *Annual Reviews in Control*, vol. 29, no. 2, pp. 273–277, 2005.

[21] C. ISO, *26262, Road vehicles–Functional safety*, Std., 2011.

[22] R. S. Kaplan and D. P. Norton, *The balanced scorecard: translating strategy into action*. Harvard Business Press, 1996.

[23] N. Mellegård, M. Staron, and F. Törner, "A light-weight defect classification scheme for embedded automotive software and its initial evaluation," in *Software Reliability Engineering (ISSRE), 2012 IEEE 23rd International Symposium on*. IEEE, 2012, pp. 261–270.

[24] 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," *Software Engineering, IEEE Transactions on*, vol. 18, no. 11, pp. 943–956, 1992.

[25] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, and F. Torner, "Evaluating long-term predictive power of standard reliability growth models on automotive systems," in *Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on*. IEEE, 2013, pp. 228–237.

[26] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, F. Törner, W. Meding, and C. Höglund, "Selecting software reliability growth models and improving their predictive accuracy using historical projects data," *Journal of Systems and Software*, vol. 98, pp. 59–78, 2014.