Assessing the release readiness of engine control software

Sichao Wen*
Volvo Car Group
sichao.wen@volvocars.com

Christoffer Nilsson
Volvo Car Group
christoffer.nilsson@volvocars.com

Miroslaw Staron
Chalmers/University of Gothenburg
miroslaw.staron@gu.se

ABSTRACT

Powertrain control calibration is an essential stage before the delivery of final products, to ensure that a vehicle works well in all driving environments. However, due to complexity of a powertrain control system (often as many as 40,000 parameters), it is difficult to assess when all parameters are fully calibrated. Therefore, the aim of this paper is to explore an approach to evaluate and predict the maturity level of powertrain control software based on calibration data for simulation models. We developed metrics that indicate software maturity and their visualization using heatmaps. We used software maturity growth curves to select maturity growth models for maturity assessment. The results and approaches of this paper were validated with theoretical methods and empirical methods using action research techniques. Our conclusions show that we can use standard software reliability growth models to monitor the calibration process for powertrain software, and we conclude that this type of monitoring provides a good support for quality stakeholders in monitoring powertrain calibration quality.

CCS CONCEPTS

• **Software and its engineering → Software reliability**;

KEYWORDS

software maturity, software metrics, data visualization, regression analysis

ACM Reference Format:
Sichao Wen, Christoffer Nilsson, and Miroslaw Staron. 2018. Assessing the release readiness of engine control software. In *SQUADE'18:: SQUADE'18:IEEE/ACM 1st International Workshop on Software Qualities and their Dependencies , May 28, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3194095.3194099

1 INTRODUCTION

The importance of software has become incredibly notable in the automotive industry nowadays. During the past decades, innovations based on software have been widely applied to make passengers feel more comfortable on board, to improve the safety of driving, to achieve better fuel efficiency for powertrain, etc. [5]. However, due to the increasing complexity of systems, various challenges appear as an accompaniment in many areas such as the process of software development, software quality management and system testing [5]. When it comes to powertrain control systems, the powertrain control calibration, which typically concerns the tuning of control algorithms, is very important for powertrain development. The vehicles equipped with well calibrated powertrain control software fulfill the specific requirements of consumption, emission, driving performance, etc. [6]. However, how to reduce the cost for powertrain control calibration has become a challenging problem for the automotive industry, because powertrain calibration is based on a time-consuming, experimental work.

In the area of software engineering, a lot of research work has been done with regard to management of software quality. For example, the research on release readiness aims at evaluating the quality of software and predicting when the software can be released [13]. Software reliability, which is defined as the probability of failure-free software operation for a specified period of time in a specified environment [1], also performs as one of the key factors to measure software quality [8]. However, in spite of the fact that release readiness and software reliability provide techniques for the evaluation of software quality, we need to customize them to use them to explain the software maturity addressed for powertrain control calibration heres. Therefore, we set off to address the following research problem:

- *How should the maturity of software be evaluated in terms of powertrain control calibration?*

We conducted an action research study at Powertrain Department of Volvo Car Corporation. We defined a metric to measure the maturity level of powertrain control software, based on the change of parameters in the control algorithm and signals in the control system. We use the metric to build a maturity model for the powertrain control software and we provide a visualization using heatmaps, which manages to visualize the calibration work and help with analysis and project management in this way. We validate this method on another project from the same company, confirming that the metric is valid and new maturity models can be constructed based on it; maturity models specific for the functionality of each project.

2 BACKGROUND

Due to the wide application of new techniques like simulation methods, model-based methods, etc. in the powetrain development, modeling and simulation are heavily utilized for the development of powertrain control systems.

In this paper, the *powertrain control system* refers to the entire control system in a vehicle powertrain. Due to the modular architecture, many sub-systems can be further defined, according to their functionalities within the overall control system [3]. In general, a powertrain control system consists of software and hardware

*This work is based on this author's master's thesis [14].

SQUADE'18:, May 28, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5737-1/18/05…$15.00
https://doi.org/10.1145/3194095.3194099

components. *Powertrain control software* is used here to specifically refer to the software part of the powertrain control system, which consists of many *software modules* corresponding to different software-related functionalities of the powertrain control system. Within each software module, there are various *models* that are designed to cooperate together and achieve relevant functionality. The term *model* here, refers to models developed with tools like Simulink by MathWorks for the purpose of modeling a real system and simulating certain phenomena to test if the desired function can be achieved.

Calibration variables in this paper come from the model elements in a powertrain control system. The value of the parameters in model elements can be a scalar, a vector, a matrix with multiple rows and columns, or a string corresponding to the name of another signal.

2.1 Powertrain Control Calibration Process

Traditionally, the process of powertrain control calibration largely relies on manual calibration work, i.e. engineers trying out different parameter configurations, to achieve the optimal engine configurations for performance, comfort, environment-friendliness, etc.

With emphasis on many different aspects regarding the functionality, a complete calibration project goes through different stages, after the selection of hardware and the setting of control strategy. These stages include base calibration, fuel consumption and emission calibration, drivability calibration and on board diagnosis calibration [11]. With the application of modern calibration methods, a general process is followed by every stage above. The maturity of powertrain control software changes significantly during this process. This general process is illustrated in Figure 1 and briefly explained as follows:

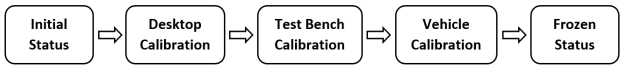

Figure 1: A general calibration process.

The initial status indicates the start of a new calibration project. The preparation work before calibration will be finished within this phase, i.e. the company has completed the selection of control hardware and finished the design of control algorithms.

The desktop calibration phase concerns the virtual calibration during which detailed calculations are done for the definition of necessary calibration parameters in simulation models [11]. Calibration engineers calculate the calibration parameters at desktop based on technical models, simulation and their previous experience. The calibration data can also be transferred from an old mature project in this phase, if possible.

In the test bench calibration phase, engines run on the test bench, which is a set-up of the electrical nodes with software in a dedicated lab environment. The calibration parameters are tested. Statistical methods like Design of Experiments are usually used to create mathematical models that manage to predict the engines outputs with fewer tests [10]. After the measurement of essential points, some extra points can be added for validation to ensure the quality of models [11]. Calibration engineers monitor the entire measuring procedure to ensure the reliability of results [11].

During the vehicle calibration phase, calibration engineers drive vehicles in all representative environments with different ambient temperatures, altitudes, markets and driving behaviors of customers taken into consideration [11]. They collect more data and further tune the calibration parameters. At the end of this phase, the functionality of the powertrain fulfills the specific requirements.

In the frozen status phase, all of the calibration data get frozen once approved, which means that the calibration work comes to an end.

3 RESEARCH DESIGN

This section details the design of our study.

3.1 Research questions

As stated in Section 1, the study is implemented based on the software part of powertrain control system, aiming to look for an efficient way of evaluating and predicting the maturity level of powertrain control software, during the whole calibration process, by analyzing historical project data. In order to achieve this final goal, the following research questions need to be addressed for our study:

Q1 *How to identify stable calibration variables before delivery milestones?*

The solution to this question offers an essential element for the evaluation of software maturity levels. In addition, the definition of stable calibration variables is also affected by the way of measuring maturity for software, which results in another question to be solved:

Q2 *How to define the maturity levels of powertrain control software with calibration data?*

The solution to this question is related to how the calibration of powertrain control software is conducted and managed. What's more, when it comes to the analysis of calibration data, the investigation on powertrain control models can help readers analyze the mode of calibration work, which is taken into consideration as well. Hence, the third research question is raised:

Q3 *How to make use of the structure of the powertrain control system during the evaluation and prediction of software maturity?*

By linking calibration variables to the structure at different levels, including the signal or block level, the subsystem level, the model level and the final system level, it is possible to explore how the calibration work is done as well as how the maturity of software changes during the project cycle. Since the building of maturity model aims to help with planning and management work, it is necessary to make the results intuitive to stakeholders, hence the next research question:

Q4 *What visualization methods are suitable for different stakeholders?*

Different stakeholders can be managers, developers, calibration engineers, and so on. According to different needs of them, the optimal visualization method may vary.

3.2 Data Selection

For the development and validation of our new metric and model, the calibration project which is selected for data collection should have been finished, so that the entire cycle during which the software maturity changes from zero to one hundred percent are available for analysis. The history of the selected calibration project should be as complete as possible, which means the calibration variables and the attributes of them corresponding to every phase of the project should be stored in data sources, so that it is possible for the researchers to make the best use of calibration data.

Table 1 presents an example of the data used for the analysis of calibration variable *var_x* after data selection, which includes how the value and the maturity measure value changed for *var_x*, as well as when the change happened. It shows a complete calibration process for *var_x*. V_i represents the value of *var_x* and Q_i represents the maturity of *var_x* measured by internal standards at Volvo Car Corporation. The calibration variable *var_x* was in the simulation model *model_x*. Every time a change of value was committed, V_i was updated to V_{i+1}. Similarly, every time the maturity of *var_x* changed, Q_i was updated to Q_{i+1}.

Table 1: Example data that will be used for the analysis of calibration variables.

Name	Value	Update Date	Maturity Measure	Model
var_x	V_1	week 1	Q_1	model_x
var_x	V_2	week 3	Q_2	model_x
var_x	V_2	week 10	Q_3	model_x
var_x	V_2	week 20	Q_4	model_x
var_x	V_3	week 25	Q_4	model_x
var_x	V_3	week 52	Q_5	model_x

3.3 Data Processing

The values of calibration variables are the basis of maturity analysis. However, the values vary in data types. Some of them are numerical data, including scalars, vectors and matrices with more than one row, as well as more than one column. Some of them are strings which indicate the names of other signals in the system. For the sake of consistency, the values are converted, so that they share the same data type. Instead of the values themselves, the update frequency of values is counted based on the calibration history, which is similar to the approach for the analysis of code stability in [12]. Then, not only will the data types be consistent as scalars but also the management of calibration concerning efficiency will be benefited.

The calibration history of a calibration variable is traced with

$$\boldsymbol{s} = \{s[n]\},\ 0 \leqslant n \leqslant l-1, \tag{1}$$

where $s[n]$ is the number of changes that happened in n_{th} week, and l is the number of weeks that have been spent on the calibration project. Considering that calibration variables can be labeled by the model(s) to which they belong, we can group the calibration variables by models and summarize the total number of changes within each model in the following way:

$$\boldsymbol{m}_k = \sum_{s_{ki} \in C_k} \boldsymbol{s}_{ki}. \tag{2}$$

Here, C_k is a set consisting of all calibration variables that belong to model k.

3.4 Visualization

A good tool that makes the calibration management efficient desires appropriate visualization of the data given above. However, due to large quantities of data in a system, a wise visualization method must be applied. It has been verified that heatmaps have a good performance on the visualization of large quantities of data [12]. The basic idea is that we can use color information to present the value of $s[n]$ and $m[n]$. In this way, it is possible to visualize a large number of $\boldsymbol{s}$'s and $\boldsymbol{m}$'s by plotting only once, and it will be much easier to compare different variables and help manage the calibration of different variables since colormaps lead to intuitive understanding for human beings. In addition, heatmaps make the correlation among different variables much easier to be noticed.

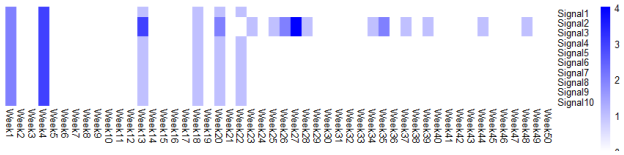

Figure 2: An example of visualization using heatmaps, where the rows show a high correlation.

Figure 2 shows an example of visualization using heatmaps. Each row corresponds to a $\boldsymbol{s}$ or $\boldsymbol{m}$ and each column corresponds to a week. As indicated in the legend, darker blue means a larger value for the corresponding point. If we consider columns as objects to be compared, we can notice correlation which corresponds to different milestones during a calibration process. For the example in Figure 2, there are several blue columns which indicate the corresponding weeks can be special time points in the calibration process, while the other white or mostly-white columns may be less interesting from management point of view. Based on the analysis above, heatmaps offer readers a general impression of the correlation among variables.

3.5 Software Maturity Evaluation

In this section, metrics for the measurement of software maturity are defined in respect of powertrain control calibration. Then, the modeling of the software maturity growth curve is investigated in order to predict the maturity level in the future. Finally, the methods to validate modeling are discussed.

3.5.1 Software Maturity Metrics. Because our study is based on model-based powertrain control software, we define the cumulative percentage of mature models as an indicator of the maturity level of powertrain control software. This indicator is theoretically valid since if more models are identified as "mature", more software-related functions within the entire system can be fulfilled, which typically indicates that powertrain control software is at a higher

mature level. This quantifiable indicator of software maturity (ISM) can be calculated as follows:

$$ISM = \frac{\#\ mature\ models}{\#\ all\ models}, \tag{3}$$

i.e. the number of mature models divided by the total number of models in software.

Now the question left is how we should identify a mature model based on calibration data. Since calibration components can be grouped by the model to which they belong, they can be considered as the attributes of the corresponding model. Then the maturity of a model can be measured by the maturity of the calibration components within this model. We adopt strict rules on the identification of mature models and use boolean values to represent and calculate the maturity of a model as well as a calibration component. We use *1* to represent "mature" and use *0* to represent "immature".

When it comes to the definition of a mature calibration component, the stability of the calibration component is checked and relevant attributes are taken into consideration. With referring to the internal measures of the maturity of calibration variables at Volvo Car Corporation, the term *stability* here is closely related to the value of a calibration variable and the calibration phase to which the calibration variable belongs. The consideration is because powertrain control calibration relies on the tuning of control algorithms, which is essentially the tuning of parameters in calibration components. Hence, the change frequency of values indicates the stability of calibration components. If the value of a calibration variable is updated frequently, the corresponding calibration component cannot be very stable. Since the change of values may happen at any phase during the calibration process unless the final frozen status is reached, there can be potential violation of stability at a comparatively early phase. Thus, it seems to be necessary to consider the calibration phase to which the calibration variable belongs for the identification of stability. Based on these two attributes, we propose three definitions for the identification of a stable calibration component:

- D1 *A calibration component is considered stable if and only if it is at the frozen status, and the date when it becomes stable is the same as the date when it reaches the frozen status.*
- D2 *A calibration component is considered stable when the value of the corresponding calibration variable is considered consistent based on the known information, and the date when it becomes stable is the date of the last change of value.*
- D3 *A calibration component is considered stable if and only if it is at the frozen status, but the date when it becomes stable is the date of the last change of value.*

All of these three definitions will be evaluated in the consecutive steps. The date when a model becomes mature is the date when the logical *and* of all the maturity of calibration components in the model changes from *0* to *1*. Therefore, although the illustration in this section follows a top-down method, the calculation of the software maturity indicator is actually based on a bottom-up method following a sequential order of identifying stable calibration components, identifying mature models and finally calculating the software maturity indicator.

3.5.2 Modeling of Software Maturity. Based on the software maturity indicator defined in Section 3.5.1, *ISM* can be plotted as a function of time t in the form of

$$ISM = f(t). \tag{4}$$

Typically, $f(t)$ is a non-linear monotonically increasing function and $0 \leq f(t) \leq 1$. The curve plotted with observations of $f(t)$ is of class C^0 but might not be of class C^1 since its first derivate $f'(t)$ is very likely to be discontinuous. However, it is possible for us to model $f(t)$ with a smooth function $f_0(t)$, since the influence of outliers or noise is usually minor for the estimation of $f_0(t)$ and the value of $f(t)$ is obtained based on large quantities of variables.

The focus of this section is to find a $f_0(t)$ that manages to estimate $f(t)$ with least squares of error when $0 \leq f(t) \leq 1$. There are a lot of statistical models which are used to measure attributes of software quality, such as those based on defect counting [2]. One example of the models based on defect counting is the software reliability growth model, which we will refer to in this paper. The reason is that mathematical models that describe the growth of software reliability, which is defined as *the probability of failure-free software operation for a specified period of time in a specified environment* [1], are proposed based on statistics and stochastic processes [16] and they can be considered as growth models in general [9]. Some representative software reliability growth models, that we investigated in our study are: (i) Goel-Okumoto model [4], (ii) Inflected S-Shaped Model, (iii) Delayed S-shaped [15], (iv) Logistic Model, and (v) Yamada's testing estimation model [16].

We choose least squares methods here to estimate the parameters in these non-linear models. Depending on the specific growth curve, $f_0(t)$ can also be a linear function and then linear regression methods will be applied. However, since $f_0(t)$ is a non-linear function in most cases, we focus on non-linear least squares here. Levenberg-Marquardt algorithm [7] is used in our study for non-linear regression. After the estimates of parameters are obtained, $f_0(t)$ can be constructed. The next step is to evaluate the quality of the constructed model in terms of the accuracy of fitting and its prediction power.

To examine the accuracy of fitting, we use root-mean-square error (RMSE) and adjusted R^2. As for the evaluation of prediction power, we can use part of observations for modeling and evaluate how well the constructed model predicts the other observation points by calculating the metrics for goodness of fit given above using predicted points as well. For example, we can use the first m observations ($1 \leq m \leq n$) for the estimation of model parameters and then apply the constructed model to the calculated n estimates of data points, and then we can calculate *RMSE* using all n estimates and n observations and choose the model that results in the smallest *RMSE*. Here, m can be either small or large for the purpose of checking prediction power at different time points. The optimal growth model can be selected based on both goodness of fit and prediction power.

4 RESULT ANALYSIS

We grouped calibration variables by models, visualized the total number of updates to values and maturity measures in each model using heatmaps, and applied agglomerative hierarchical clustering at the same time. The results are showed in Figure 3 and Figure 4. One row corresponds to one model and one column corresponds to one week, but the labels are hidden for confidentiality.

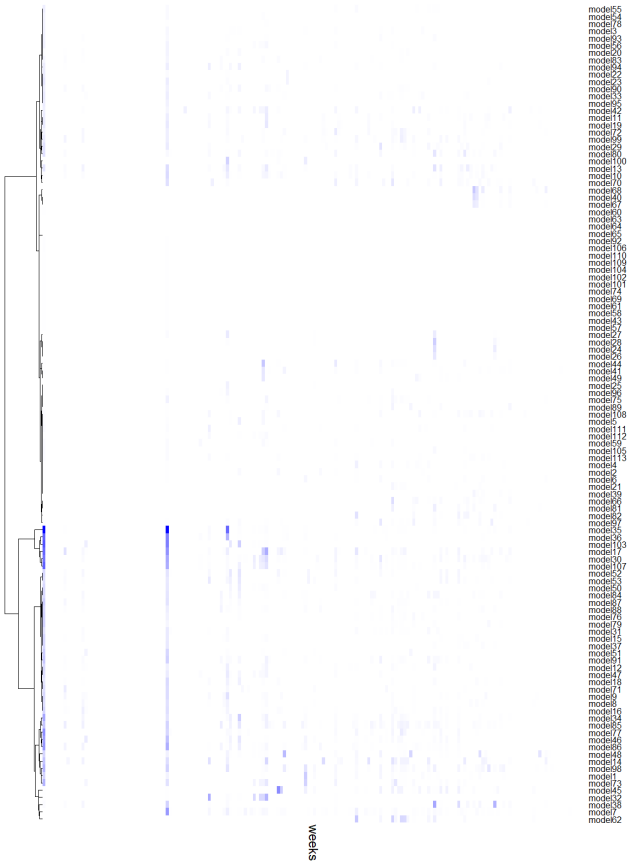

Figure 3: Visualization of the total number of updates to values in each model.

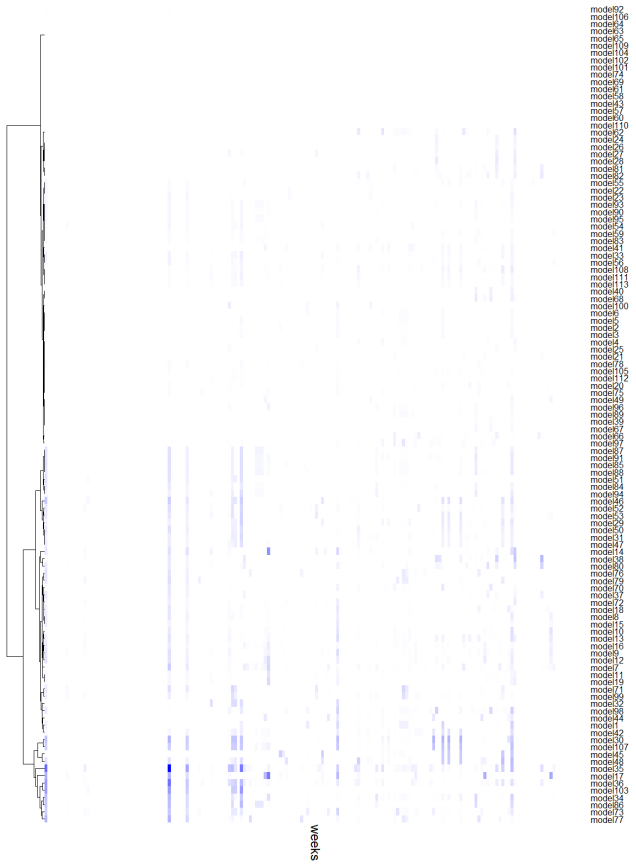

Figure 4: Visualization of the total number of updates to maturity measures in each model.

The cumulative percentage of mature models was calculated based on the definitions of stable calibration components discussed in Section 3.5.1. The three curves are presented in Figure 5. *D1* represents the curve calculated based on whether or not a calibration component has reached the frozen status. *D2* represents the curve calculated according to the change of values until the last week in the dataset. The calculation of *D3* took both aspects into consideration. The x-label performs as a timeline throughout the project, although the specific numbers are hidden. Observing the three growth curves, we find that the slopes of the three curves are all very small in the beginning. After half of the project, the maturity grows exponentially for all three definitions, while the curves become flat again near the end of the project. The shapes of curves indicate that it takes a long time for calibration efforts to be accumulated, and the effects break out at the last half of the project, which can be caused by the great complexity of powertrain control software and the correlation of different calibration variables. For example, the calibration of some variables cannot start until the calibration of some other variables has been finished. We also find that the curves of *D2* and *D3* both start from a positive number, which indicates that some values were never changed for the specific project. Hence, during parameter estimation for different models, a constant parameter was added to the mean value functions of the models in Section 3.5.2 to represent the initial maturity.

The modeling procedure followed the method introduced in Section 3.5.2. Adjusted R^2 was primarily used to judge the quality of regression and whether the estimates of one trial were reliable. To select the best-fitting model, we focused on the prediction power of the models and used *RMSE* to judge the accuracy of prediction. We used the first 50%, 60%, 80% and 100% of observations respectively for the construction of models, and then used all estimates to calculate *RMSE*. Comprehensively considering the performance of models for these four tests, we were able to determine the best-fitting model. Because parameters in models were initialized randomly, the initial guess of these parameters can be really bad sometimes. In this case, the estimates of parameters may not converge within the maximum number of iterations, and there can also be some problems with certain matrix calculations. So sometimes the non-linear least squares algorithm failed or just gave inaccurate estimates. Multiple trials were implemented in order to obtain the optimal estimates for each model.

Take the modeling of curve *D3* for example. The *RMSE* we got for each model is compared in Figure 6. In general, the prediction error decreases as more observations are used for modeling. We can

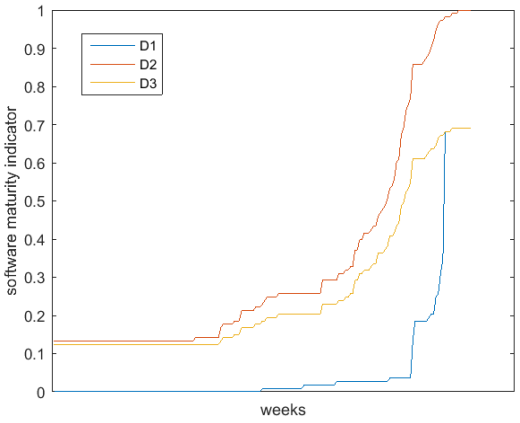

Figure 5: The growth curves of cumulative percentage of mature models, i.e. software maturity indicator, based on three definitions of stable calibration components.

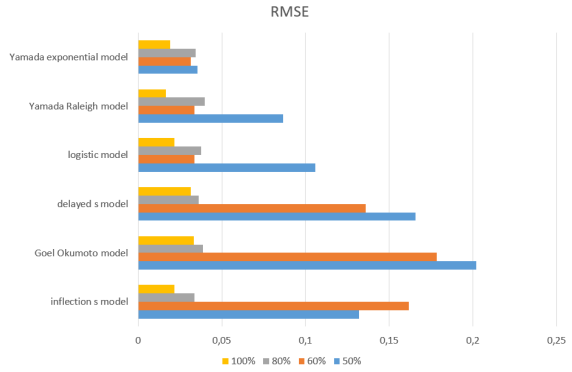

Figure 6: RMSE of six models for Definition *D3*. The first 50%, 60%, 80% and 100% of observations were used respectively.

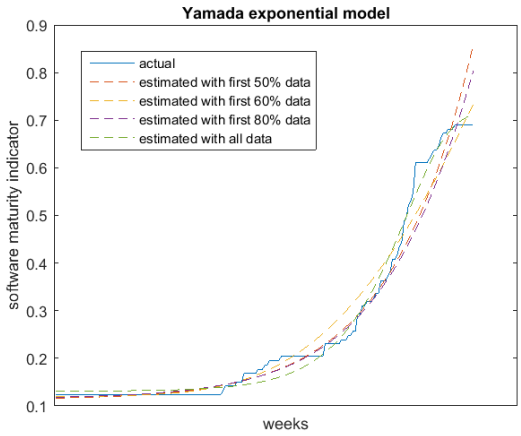

Figure 7: Modeling of the growth curve based on Definition *D3* using Yamada Exponential Model.

see that Yamada exponential model has the optimal performance, because it is very robust and achieves the minimum error in almost every case. The modeling result of Yamada exponential model is showed in Figure 7. From Figure 7, we find that the constructed model fits the curve very well in all cases. The largest prediction error happens at the end of the project, when the software reaches the highest maturity and the trend of growth is no longer exponential. In spite of the deviation near the highest maturity, the constructed model predicts the software maturity growth curve of Definition *D3* very well in general, which indicates the necessity of test-effort consideration.

Similarly, we chose the best-fitting model for the other two definitions of metrics, and the corresponding results are showed in Figure 8. Logistic model was chosen for both cases. The modeling of Definition *D2* looks good in general. Although the predictions based on the first 50% and 80% of observations are a bit more biased than that based on the first 60% observations, the results are generally acceptable. This means that Logistic Model is suitable for describing the trend of maturity growth if only the change of the values of calibration variables is considered. However, the modeling of Definition *D1* doesn't seem very accurate. In this case, the original curve looks more similar to a step function, which makes it hard to predict the trend of the curve based on the first half of observations, since the curve is almost flat at the early stage of the project. Actually, instead of predicting based on growth models, it may be more reliable to calculate this curve based on the planning of calibration projects regarding important milestones, such as when a calibration phase should be finished and when the next stage should start.

4.1 Validation of Results

In order to validate our results, we replicated the study on another, similar project in the same organization, following the same analytical procedures. The software maturity growth curves of the new project are shown in Figure 9. They are different from the apparent concave, upward functions in the first project. The software maturity growth curves in the new project seem to have approximately constant slopes. Definition *D2* and *D3* give two curves that grow almost linearly instead of exponentially, while Definition *D1* still gives a curve that looks like a step function. Through the interviews with the reference group, we found that the two projects were initialized with different status. The second project started following the first project. Large quantities of data and models were transferred to it from the previous project. Since the models had been well calibrated in the previous project, different calibration efforts may be possible. This indicates that it is unreliable to directly use the matemathical growth models constructed in the first project for the second one. However, using the modeling approaches in Section 3.5.2, we managed to construct models with good predictive power for Definition *D2* and *D3*, although it is still hard to predict the step function of Definition *D1* at an early stage, which means that the approach is still valid. See Figure 10, Figure 11 and Figure 12.

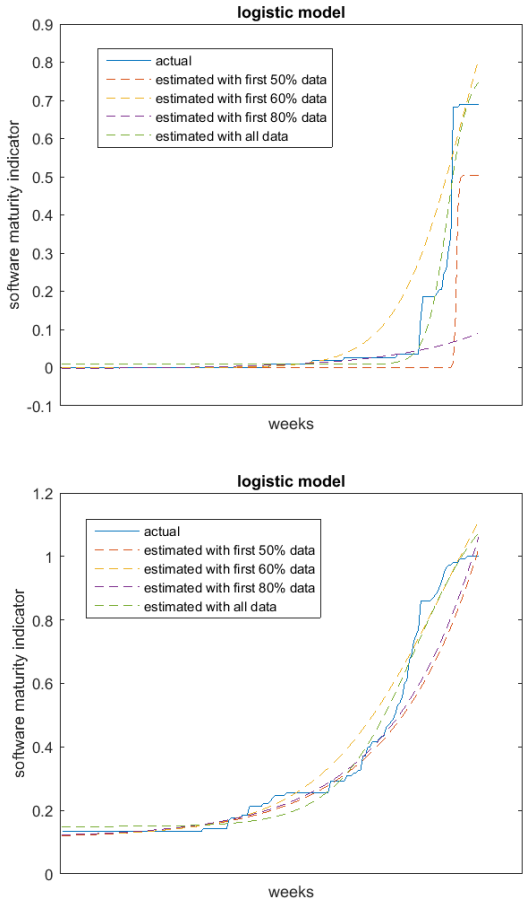

Figure 8: Modeling of the growth curve using Logistic Model. From top to bottom, the first figure is for Definition *D1*, and the second figure is for Definition *D2*.

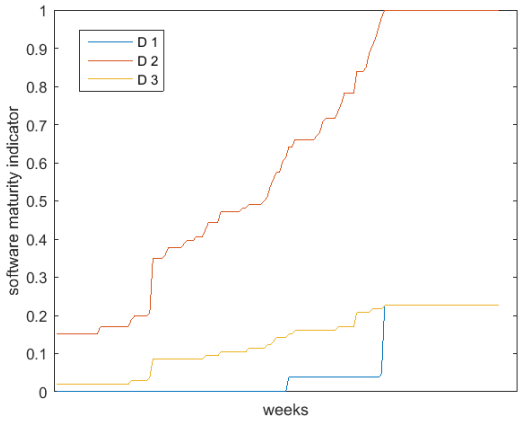

Figure 9: The growth curves of cumulative percentage of mature models in the new project, based on three definitions of stable calibration components.

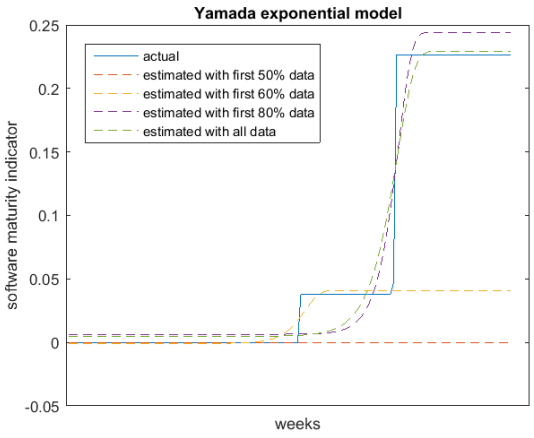

Figure 10: Modeling of growth curves for the new project using Yamada Exponential Model. Definition *D1*.

With grouping calibration variables per model and calculating the cumulative percentage of mature models, we manage to monitor the maturity level of software and predict the maturity by statistical models for different projects.

5 CONCLUSIONS

When developing engine control software for powertrain systems, an important quality aspect is the ability to fine-tune the parameters for optimal drivability, performance, economy, environment or custom type of driving. Vehicle manufacturers often can use as many as 40,000 calibration parameters, which makes the calibration process time-consuming. In this process, it is important to be able to track when the software is ready to release, i.e. the powertrain is ready calibrated. Efficient management of powertrain control calibration requires the visualization, evaluation and prediction of software maturity.

In this paper, we developed new metrics to measure the maturity level of powertrain control software for model-based software development. We also complemented these measures with the visualization method and validated them on another project. Our validation shows that the formal models need to be adjusted from project to project, but the definition of the maturity metric is the same.

Our study helps to develop a new measurement system for model-based powertrain control software, to evaluate and predict software maturity. Our further work is to evaluate the metric on more projects.

REFERENCES

[1] 1990. IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990* (Dec 1990), 1–84. https://doi.org/10.1109/IEEESTD.1990.101064

[2] Vard Antinyan, Miroslaw Staron, Anna Sandberg, and Jörgen Hansson. 2016. Validating Software Measures Using Action Research a Method and Industrial Experiences. In *Proceedings of the 20th International Conference on Evaluation*

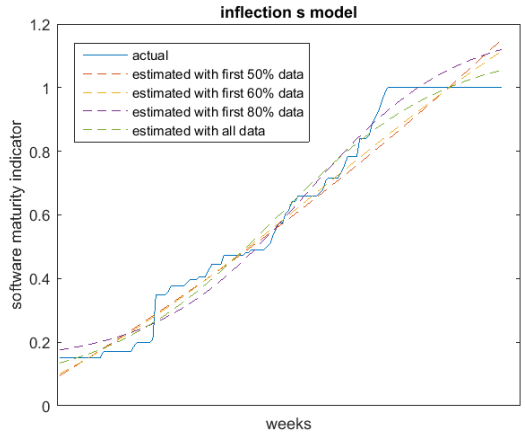

Figure 11: Modeling of growth curves for the new project using Inflection S Model. Definition *D2*.

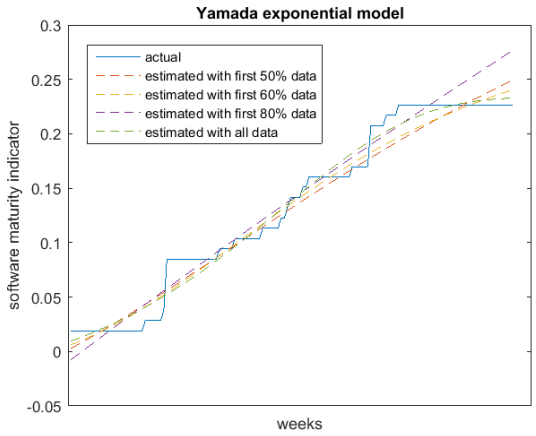

Figure 12: Modeling of growth curves for the new project using Yamada Exponential Model. Definition *D3*.

and Assessment in Software Engineering (EASE '16). ACM, New York, NY, USA, Article 23, 10 pages. https://doi.org/10.1145/2915970.2916001

[3] Dauron, A. 2007. Model-Based Powertrain Control: Many Uses, No Abuse. *Oil & Gas Science and Technology - Rev. IFP* 62, 4 (2007), 427–435. https://doi.org/10.2516/ogst:2007054

[4] A. L. Goel and K. Okumoto. 1979. Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures. *IEEE Transactions on Reliability* R-28, 3 (Aug 1979), 206–211. https://doi.org/10.1109/TR.1979.5220566

[5] Klaus Grimm. 2003. Software Technology in an Automotive Company: Major Challenges. In *Proceedings of the 25th International Conference on Software Engineering (ICSE '03)*. IEEE Computer Society, Washington, DC, USA, 498–503. http://dl.acm.org/citation.cfm?id=776816.776878

[6] Keith Lang, Michael Kropinski, and Tim Foster. 2010. Virtual Powertrain Calibration at GM Becomes a Reality, In SAE Technical Paper. https://doi.org/10.4271/2010-01-2323

[7] Manolis IA Lourakis. 2005. A brief description of the Levenberg-Marquardt algorithm implemented by levmar. *Foundation of Research and Technology* 4, 1 (2005).

[8] Michael R. Lyu (Ed.). 1996. *Handbook of Software Reliability Engineering*. McGraw-Hill, Inc., Hightstown, NJ, USA.

[9] Ganesh J. Pai. 2013. A Survey of Software Reliability Models. *CoRR* abs/1304.4539 (2013). http://arxiv.org/abs/1304.4539

[10] Eric Rask and Mark Sellnau. 2004. Simulation-Based Engine Calibration: Tools, Techniques, and Applications, In SAE Technical Paper. https://doi.org/10.4271/2004-01-1264

[11] Axel Schlosser, Bert Kinoo, Wolfgang Salber, Sascha Werner, and Norbert Ademes. 2006. Accelerated Powertrain Development Through Model Based Calibration, In SAE Technical Paper. https://doi.org/10.4271/2006-01-0858

[12] M. Staron, J. Hansson, R. Feldt, A. Henriksson, W. Meding, S. Nilsson, and C. HÃűglund. 2013. Measuring and Visualizing Code Stability – A Case Study at Three Companies. In *2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement*. 191–200. https://doi.org/10.1109/IWSM-Mensura.2013.35

[13] Miroslaw Staron, Wilhelm Meding, and Klas Palm. 2012. *Release Readiness Indicator for Mature Agile and Lean Software Development Projects*. Springer Berlin Heidelberg, Berlin, Heidelberg, 93–107. https://doi.org/10.1007/978-3-642-30350-0_7

[14] Sichao Wen. 2017. *Understanding and visualizing software maturity based on calibration parameters in powertrain control software*. Master's thesis.

[15] Alan Wood. 1996. Software reliability growth models. *Tandem technical report* 96, 130056 (1996).

[16] Shigeru Yamada. 2013. *Software Reliability Modeling: Fundamentals and Applications*. Springer Publishing Company, Incorporated.