Predicting Fault Inflow in Highly Iterative Software Development Processes: An Industrial Evaluation

Martin Bäumer, Patrick Seidler, Richard Torkar and Robert Feldt
Blekinge Institute of Technology, Sweden
Corresponding author: *martin.baeumer@gmx.net*

Piotr Tomaszewski and Lars-Ola Damm
Ericsson AB, Sweden
{*piotr.tomaszewski*|*lars-ola.damm*}*@ericsson.com*

Abstract

This paper addresses the need for accurate predictions on the fault inflow, i.e. the number of faults found in the consecutive project weeks, in highly iterative processes. In such processes, in contrast to waterfall-like processes, fault repair and development of new features run almost in parallel. Given accurate predictions on fault inflow, managers could dynamically re-allocate resources between these different tasks in a more adequate way. Furthermore, managers could react with process improvements when the expected fault inflow is higher than desired. This study suggests software reliability growth models (SRGMs) for predicting fault inflow. Originally developed for traditional processes, the performance of these models in highly iterative processes is investigated. Additionally, a simple linear model is developed and compared to the SRGMs. The paper provides results from applying these models on fault data from three different industrial projects. One of the key findings of this study is that some SRGMs are applicable for predicting fault inflow in highly iterative processes. Moreover, the results show that the simple linear model represents a valid alternative to the SRGMs, as it provides reasonably accurate predictions and performs better in many cases.

1. Introduction

An essential characteristic of highly iterative processes is the development of new features running almost in parallel to fault repair [3]. At the beginning of a project, project management allocates resources in terms of man-hours to both of these activities. The number of faults that have to be repaired play a major role for managing those resources [16]. An unexpectedly large number of faults poses challenges to resource planning and, thus, to meet project goals. Either additional man-hours have to be provided or existing resources have to be reallocated. In the former case, the project might exceed the estimated costs. In the latter case, development might not be able to deliver all the planned functionality, because part of the resources are used for fault removal instead.

An important measure to support management in project planning is the fault inflow. This fault inflow is the number of faults measured during several consecutive weeks within a project, i.e. the distribution of faults over a period of time during a project. Predicting fault inflow would enable management to plan for required resources in advance and to dynamically reallocate them. Furthermore, managers could introduce process and quality improvements beforehand in case the inflow of faults is predicted to be higher than expected. Thus, costs could be controlled more accurately and a shortage of resources can be prevented.

This study is based on fault data from three large-scale software projects at Ericsson AB. All projects follow a *highly* iterative development process. This process implies a weekly or more frequent delivery of new system versions spanning almost the complete project time, while testing begins early in the project and proceeds continuously. In such *highly* iterative contexts, information on fault inflow is particularly important for project planning to reach the project goals, as the challenges to manage iterative processes are more intensified.

The study's aim is to support management with a simple method that delivers reliable predictions on the

fault inflow at reasonable costs. To be cost-effective, the method shall be based either on existing data or on data that can be collected with low effort.

This paper suggests software reliability growth models (SRGMs) for predicting fault inflow. Particularly, this study covers the following models: Gompertz [9], Delayed S-shaped [22], Yamada exponential [23], and Goel-Okumoto [7]. These models are widely used to predict the release date of a software product. SRGMs base their predictions on data from the testing process and, thus, reflect the testing, the fault introduction and the fault finding processes. Therefore, SRGMs are, in our opinion, potentially suitable for predicting fault inflow as well. One additional benefit of SRGMs is that their use can be easily automated and supported with tools.

Based on our observations of the fault inflow, this study presents an additional linear model that appears to reflect fault inflow of highly iterative processes and is simpler than the tested SRGMs.

This study addresses the following research questions:

RQ 1 *How do the models under study perform in predicting the fault inflow for the remaining project time?*

RQ 2 *How do the models under study perform with regard to short-term predictions?*

For RQ 1, we further test the following hypotheses:

H_0: There is no significant difference between the models in their performance of fault inflow prediction for the remaining project time.

H_1: There is a significant difference between the models in the performance of fault inflow prediction for the remaining project time.

As a result, the study shows that SRGMs can be used to predict fault inflow. Additionally, it is shown that the proposed linear model delivers accurate predictions as well and thus, provides a valid alternative to SRMGs.

The remainder of this paper is structured as follows: The next section, Section 2, refers to existing research related to this study. In Section 3, this study's design is explained. Section 4 presents the results of this study. In the end, the findings of this study are discussed in Section 5, while the conclusions and future work are covered in Section 6.

2. Related work

In general, not much research addressing the prediction of fault inflow can be found. The one example that was identified is presented by Staron and Meding [16]. In their work, the authors propose a method for predicting the inflow for three weeks in advance based on project metrics and on existing fault data. This method is then compared to other estimation practices. One of their findings is that the authors' prediction method is more efficient than, for instance, expert estimations because the prediction results are more accurate and can be obtained easier.

In contrast to Staron and Meding, our study considers the use of SRGMs. Moreover, the predictions in our study are solely based on historical data as they are already available and require only little configuration. SRGMs have been deeply discussed in literature and have often been applied in research and industry alike for deciding on a release date [8]. To our knowledge, however, SRGMs have not been used for fault inflow predictions. Several authors claim that SRGMs are not applicable in practice at all as the models' assumptions are often violated [21, 6]. However, other studies have shown that SRGMs perform well in spite of such violations [17, 2].

Other approaches for predicting faults incorporate software metrics (e.g. [14, 13]) or inspection data (e.g. [18]). Besides, current research also investigates the use of Bayesian networks. Such networks are, for instance, used as support for expert estimations [4] and to define relations between software metrics [15]. These approaches, however, will not be covered in this study as they require different data that are either hard to collect or not available. Moreover, such approaches aim at predicting the total number of faults while our study addresses the prediction of fault inflow.

3. Method

3.1. Context

This study investigates three projects carried out by Ericsson AB. Hereafter, the projects are denoted as $P1$, $P2$ and $P3$. The projects are targeted towards releases of three systems that have been on the market for several years. In this time, a number of releases have been developed for each system. The systems are similarly large, i.e. approximately half a million lines of code.

All projects under study follow a *highly* iterative process. This means that development and testing activities are carried out in short iterations. Within such an iteration, a new system version, containing new functionality and fixes of previously discovered faults, is delivered to test. These deliveries occur on weekly basis or even more frequently, while testing of the new release proceeds continuously.

The faults found during testing are reported to a database and thus, fault data can be used for further analysis.

3.2. Data

The models covered in this study are applied on historical fault data. A software fault can be defined as 'a manifestation of an error in software' [1]. The fault inflow represents the number of faults found in several consecutive project weeks.

The data was collected based on the testing process and grouped by week. For instance, $P1$ lasted 26 weeks, whereas $P2$ and $P3$ lasted 30 weeks and 33 weeks respectively. That is, the fault data used in this study comprise 26, 30 and 33 weeks.

The faults are marked with priority levels A, B and C. Faults of level A are of high priority, B of medium and faults of level C have low priority. It was decided that the study only accounts for faults of priority A and B, as those invoke the highest costs for the customer and for Ericsson AB when they appear after release. The number of faults was multiplied by a random factor and thus, the values presented in this paper do not represent the real amount of faults.

The fault data was accumulated for each week using calendar time. Many authors claim that other time measurements such as execution time are more accurate as the test effort is likely to be asynchronous [12, 20]. However, data to assess the test effort directly would require a more extensive data collection, because industry does rarely store execution time. Besides, it has been shown that SRGMs perform well based on calendar time [17].

3.3. Model building

The purpose of this study was to find a method for predicting fault inflow in highly iterative processes. SRGMs were chosen as a possible tool as they can be applied cost-effectively on available fault data. In contrast to the widely spread use of SRGMs, i.e. predicting a release date for a software product, we want to observe how they perform in predicting fault inflow. A list of the models that were used in this study can be found in Table 1. These SRGMs are a synopsis of a list of models that was provided by Wood [20]. They can be further categorized into S-shaped and concave models respectively, which relates to the general outlook of their curves. In particular, the curves of concave models bend downwards, while curves of S-shaped models first converge and become concave later [20].

We applied the SRGMs on fault data from the projects $P1$, $P2$ and $P3$. The results showed a generally good performance of the models. However, we observed that in the projects under study, the fault inflow appears to be fairly constant until release. This means, the curve of cumulative faults does almost not flatten until the release date. In other words, the curve of cumulative fault inflow seems to increase linearly throughout the project. Hence, we developed our own model (see Table 1). This model is a linear model which we believe better describes the fault inflow of our environment.

The linear model was applied on exactly the same data as the SRGMs in order to compare all candidates. In particular, this study investigated the accuracy of the different models in the context of different projects.

For the prediction, both the SRGMs and the linear model are fitted to historical fault data using the least-squares method [11]. In general, estimates of the models' parameters from data can be obtained through the maximum likelihood estimation or regression methods. Maximum likelihood methods statistically provide the best approach to find estimated parameters for large sample sizes [20]. The methods solve a number of equations simultaneously to estimate the parameters numerically. However, due to the numerical approach, those methods are also complex.

In this study, the least-squares regression method is used because this is generally accepted to be the best method for small- and medium-sized samples [11]. Each model is applied to historical fault data and by choosing the correct parameters the squared error, i.e. the difference between a model's curve and the data, is minimized. The result is a model that is fitted to the available data. This fitted model can then be used to build the prediction for the fault inflow.

For instance, a model's performance is analyzed after week 10 of a 20 weeks' project. That is, all fault data until week 10 are available. The least-squares method is used to fit the model on the first 10 weeks, which results in a fitted model based on the known fault data. Next, this fitted model is evaluated on the following weeks until the release date, i.e. week 20 in this example.

3.4. Model evaluation

A model's performance is evaluated based on how much its curve differs from the curve of the actual fault inflow. For this purpose, the mean magnitude relative error (MMRE) is calculated as shown in Eq. (1) [10].

$$\mathrm{MMRE} = \frac{\sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i}}{n} \tag{1}$$

Table 1. Models used in this study: SRGMs and the linear model

Model	Shape	Structure	Equation	Reference
Gompertz	S-Shaped	Trend	$a(b^{c^t}), a \geq 0, 0 \leq b \geq 1, c > 0$	Kececioglu [9]
Delayed S-shaped	S-Shaped	NHPP	$a(1-(1+bt)e^{-bt}), a \geq 0, b > 0$	Yamada et al. [22]
Yamada exponential	Concave	NHPP	$a(1-e^{-bc^{(1-e^{(-dt)})}}), a \geq 0, bc > 0, d > 0$	Yamada et al. [23]
Goel-Okumoto	Concave	NHPP	$a(1-e^{-bt}), a \geq 0, b > 0$	Goel and Okumoto [7]
Linear Model	linear	Trend	$a+bt$	

where y_i is the actual cumulative number of faults in week i, $\hat{y}_i$ is the predicted number of faults and n is the total number of observations, i.e. test weeks. A high value for the MMRE indicates that a model's prediction differs much from the actual fault inflow.

The research questions posed in Section 1 address (a) the models' performance for the remaining project time, and (b) the performance regarding the short-term predictions.

In the first step, the models are compared based on the MMRE, which measures how much the models' predictions differ from the actual fault inflow. The models are built based on the data at 40%, 60% and 80% of the total project time. Next, the models are compared based on the accuracy of their predictions for the remaining project time. In that way, it is also possible to determine at which time in a project reliable predictions can be obtained. Additionally, a statistical test is used to identify significant differences of the models' performance.

In the second step, this study investigates how well the models predict the short-term fault inflow. Here, the MMRE is calculated for one, two and three weeks in advance.

4. Results and analysis

4.1. Overall model performance

As described in Section 3.4, the models are compared using the MMRE. The results for the MMRE are shown in Table 2. Additionally, Figs. 1–3 visualize the results of the comparison.

Fig. 1 presents the models' curves compared to the curve of the actual fault inflow at 40%, 60% and 80% of the project time in $P1$ (see subfigures 1(a)–1(c) respectively). At 40% of the project time, the linear model overestimates the fault inflow. The concave models overestimate as well, but they are closer to the actual fault inflow, especially at the end of the project. The S-shaped models demonstrate better performance, but tend to underpredict towards the end of the project. At 60% of the project time, the linear model shows a good fit to the actual fault inflow, while all other models underestimate. At 80% of the project time, all models underpredict the fault inflow, but the linear model still shows the best fit.

The MMRE (see Table 2) confirms the results from Fig. 1. At 40% in $P1$, the linear model performs worse (MMRE = 28.9%) than the S-shaped models such as the Gompertz model (MMRE = 16.0%) and the Delayed S-shaped model (MMRE = 18.7%). However, at 60% and 80%, the S-shaped models do not perform well whereas the linear model performs best. As the MMRE indicates, the concave models show a stable performance at 40%, 60% and 80% of the project time respectively (MMRE $\approx$ 20%).

With respect to $P2$, the curves of the concave models and the linear model fit similarly well to the actual inflow, while the S-shaped models tend to overestimate or underestimate respectively (see Fig. 2).

As shown in Table 2, the MMREs for the concave models and the linear model are equal at approximately 2.5% whereas the MMREs for the S-shaped models are comparatively high.

The results for $P3$ are somewhat similar to the ones of $P1$ as shown in Fig. 3. At 40% of the project time, the linear model and the concave models overestimate, while the S-shaped models underestimate the fault inflow towards the end of the project. Table 2 indicates that the S-shaped and concave models show a similar performance, while the linear model performs comparatively bad. At later times in the projects, the S-shaped models show a generally worse performance. At 60% of project time, the concave models perform best, and at 80% project stage, the linear model performs best.

The average MMRE over all projects (see Table 2) shows that the S-shaped models deliver the least accurate predictions. The concave models deliver the most stable predictions as they perform comparatively well

Table 2. Model performance in long-term fault inflow prediction based on MMRE (in %)

	40% project time					60% project time					80% project time				
	Gom	Del	Yam	G-O	Lin	Gom	Del	Yam	G-O	Lin	Gom	Del	Yam	G-O	Lin
*P*1	16.0	18.7	20.8	20.8	28.9	26.9	28.5	19.6	19.7	6.4	24.0	28.8	20.3	20.7	9.1
*P*2	27.2	17.8	2.1	2.1	2.1	22.8	10.6	2.5	2.4	2.5	10.3	7.4	2.8	2.7	2.8
*P*3	11.6	9.4	10.8	14.2	25.7	20.5	17.5	5.7	5.7	9.2	18.9	19.0	9.2	9.2	5.6
Avg.	**18.3**	**15.3**	**11.2**	**12.4**	**18.9**	**23.4**	**18.9**	**9.3**	**9.3**	**6.0**	**17.7**	**18.4**	**10.7**	**10.9**	**5.8**

Table 3. Model performance in short-term fault inflow predictions based on MMRE (in %)

	1 week prediction					2 weeks prediction					3 weeks prediction				
	Gom	Del	Yam	G-O	Lin	Gom	Del	Yam	G-O	Lin	Gom	Del	Yam	G-O	Lin
*P*1	12.5	11.5	10.2	10.3	10.9	14.5	12.5	12.0	12.1	12.5	18.8	15.8	16.4	16.6	14.4
*P*2	4.7	5.1	2.7	2.7	2.7	6.1	7.1	2.7	2.7	2.7	10.9	7.9	3.4	3.3	3.4
*P*3	5.6	5.2	2.8	3.1	10.0	6.8	6.2	3.4	3.8	10.2	9.2	8.5	3.1	3.6	10.5
Avg.	**7.6**	**7.2**	**5.3**	**5.4**	**7.9**	**9.1**	**8.6**	**6.0**	**6.2**	**8.5**	**13.0**	**10.7**	**7.6**	**7.8**	**10.4**

at 40%, 60% and 80% of the project time and with a similar MMRE. The linear model tends to highly overestimate at 40% of the project time and, thus, delivers the worst results. However, the linear model performs best at 60% and 80% of the project time.

4.2. Statistical analysis

In order to give evidence on which of the models performs significantly better, a statistical test is used. In particular, the purpose is to determine, if there are significant differences between the models' predicted values. The Kolmogorov-Smirnov test indicates that the available data is normally distributed and therefore, the paired t-test was chosen to determine whether the models' predictions differ significantly.

The paired t-test is conducted by pair-wise comparison of the models' predictions in each project at 40%, 60% and 80% of the project time. The significance level is set to $\alpha = 5\%$ ($p < 0.05$).

In comparison to the S-shaped models, the linear model and the concave models provide significantly better predictions throughout all projects at 60% and 80% respectively. At 40%, on the contrary, the results are varying. That is, the S-shaped models perform significantly better than the concave models as well as the linear model in *P*1 and *P*3 respectively. With respect to *P*2, however, the S-shaped models perform significantly worse.

The comparison between the linear model and the concave models does not reveal a particular tendency. For instance, at 40% in *P*1, the linear model performs significantly worse than the concave models. However, at 60% and 80% respectively, it performs significantly better. In *P*2, the linear model does not differ significantly from the concave models at any of the project times. With respect to *P*3, the concave models are significantly better than the linear model at 40% and 60%, while the linear model is significantly more precise at 80%.

In summary, H_0 as stated in Section 1 can be rejected as there are significant differences, especially regarding the S-shaped models.

4.3. Short-term predictions

Table 3 presents the models' performance with respect to short-term predictions.

Here, the MMRE refers to predictions made for one, two and three weeks in advance. Each MMRE is calculated as an average MMRE over all project stages of 40%, 60% and 80%.

In *P*1, the Gompertz model performs worst with an MMRE of 12.5%, 14.5% and 18.8% for all predictions of one week, two weeks and three weeks in advance. For one week and two weeks in advance, the concave models perform best, while the linear model shows the best performance for the prediction of three weeks in advance.

Analogous to Section 4.1, the concave models and the linear model show similar performance in *P*2 and the S-shaped models perform comparatively bad. In

*P*3, for all predictions the concave models perform best, while the linear model shows the worst results.

The average performance shows that the concave models deliver the best results for all predictions of one week, two weeks and three weeks in advance. The S-shaped models on average perform worse for all predictions. The linear model shows a similar good performance as the concave models for *P*1 and *P*2. However, due to the bad performance in *P*3, the linear model has the worst average performance.

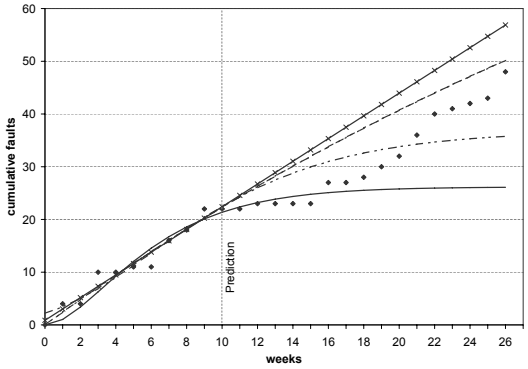
5. Discussion

5.1. Overall predictions

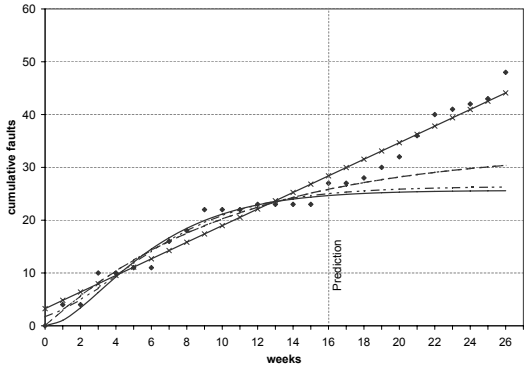
The results of this study are promising. For the overall prediction, we show that SRGMs can be used for fault inflow prediction in highly iterative software development processes. In our opinion, they show good performance for fault inflow prediction because the models' predictions are based on data from the testing process and therefore, reflect the testing process itself. However, until now the SRGMs were rather used to determine a release date.

The results of this study further indicate that the linear model performs similarly well compared to the concave models. Furthermore, the concave models and the linear model perform significantly better than the S-shaped models at 60% and 80% of the project time. That is, the concave models and the linear model are reasonably more accurate for predicting fault inflow in highly iterative processes. However, the study also shows that the models are least accurate at 40% of the project time. In our view, the reason for this observation is, that early in the project, historical data is less substantial than in later stages.

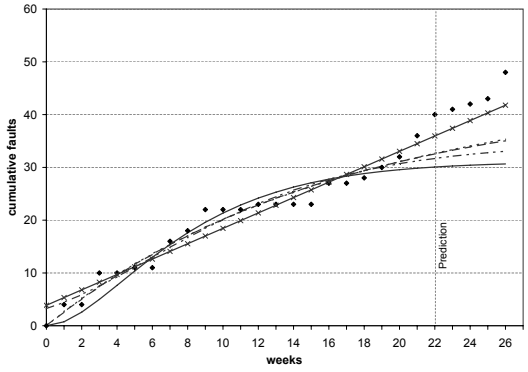
In this study, the *S-shaped models* perform worse compared to the concave models as well as the linear model, especially when measured at 60% and 80% of the project time. Additionally, they show unpredictable behavior specifically at 40% of the project time. A potential reason for that is the S-shaped character of these models. Such an S-shape reflects a learning phase within the testing process [9, 22], i.e. the test efficiency should vary strongly during testing. This, however, assumes that testing can be conducted on the complete release on which a learning process can be based on. Nevertheless, the test efficiency in the projects under study does not vary, as new functionality is added continuously. Thus, the curve of the actual faults does not become S-shaped. Another explanation is that the assumptions made by the S-shaped models are violated to such an extent that those kind of models become



(a) Project 1—Prediction at 40% project time.



(b) Project 1—Prediction at 60% project time.



(c) Project 1—Prediction at 80% project time.

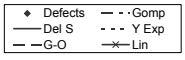


Figure 1. Project 1—Predictions at 40%, 60% and 80% of project time

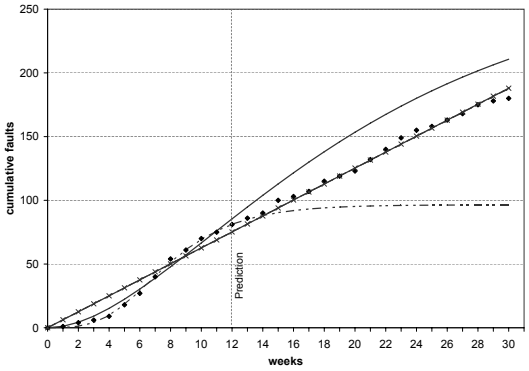
least accurate in the context of this study. For instance, the Delayed S-shaped model [22] assumes that detected faults are repaired immediately and without introducing new faults, which is not realistic in practice.

The *concave models* show the most stable performance at 40%, 60% and 80%. Still, they are not always performing best. For instance, they perform worse in some cases at 40% of the project time. An explanation is that software development projects differ in such a way that one particular model might not be suitable in all cases [12]. In other words, there does not exist a silver bullet for each scenario. However, the concave models perform significantly better than the S-shaped models at 60% and 80% of the project time. Moreover, they still perform well even though the curve of cumulated faults shows almost no flattening of the curve of cumulated faults until the late project stages. Furthermore, the performance of the concave SRGMs indicates that such concave models are applicable in practice even when the models' assumptions are violated. Thus, the results of our study confirm the conclusion that was drawn in [17] and [2]. However, although the projects for example in [2] clearly follow iterative development processes, the activities occur more sequentially, i.e. their projects base on less iterative processes than in our study. This indicates, that especially concave models are applicable in various environments.

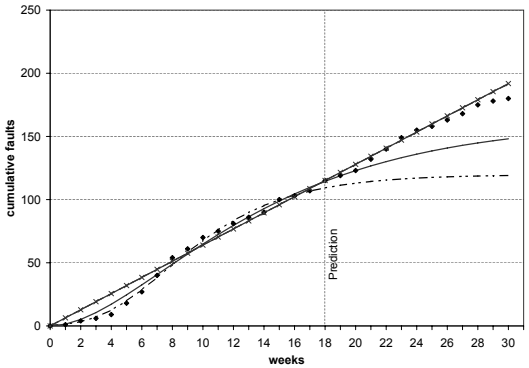
The *linear model* performs worst at 40%, but best at 60% and 80% of the project time. The performance at 40% of the project time is not surprising as the curve of cumulative fault inflow is not fully linear but slightly flattens towards the end of the project. The reason for this flattening is that less faults are found. Therefore, the linear model tends to overpredict when applied early in a project.

An explanation for the performance of the linear model at 60% and 80% of the project time is that functionality is continuously added, which in turn continuously increases the number of faults. Based on the results, we believe that the linear model performs especially well the shorter the iterations become, i.e. if new versions of the software are delivered to testing on weekly basis or even more often. In such a case, the curve of cumulative faults does not flatten throughout the project time, but remains nearly linear.

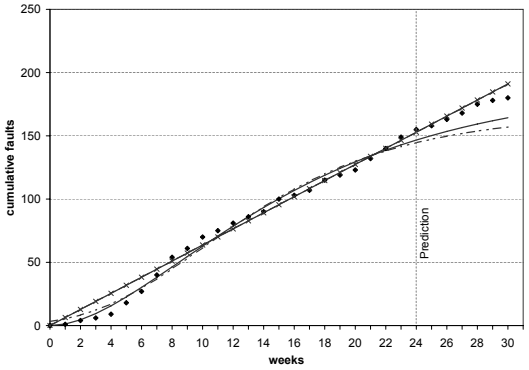
The concave models and the linear model perform equally well, i.e. with no significant difference. Nevertheless, SRGMs require a certain mathematical background and hence, practitioners might be reluctant to use such models [12]. In our opinion, the linear model presented in this study requires less mathematical background than the SRGMs under study. Furthermore, it makes no unrealistic assumptions regarding the avail-



(a) Project 2—Prediction at 40% project time.



(b) Project 2—Prediction at 60% project time.



(c) Project 2—Prediction at 80% project time.

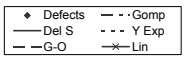


Figure 2. Project 2—Predictions at 40%, 60% and 80% of project time

able data. Consequently, the linear model is simpler to use.

In summary, the linear model represents a valid alternative for predicting the fault inflow in highly iterative software development processes. However, for situations where concave models can be accepted, or 'hidden' in tools, they can potentially give better early fault inflow prediction performance.

5.2. Short-term predictions

In addition to the overall performance in the prediction, this study investigated the ability of SRGMs and the linear model to make short-term predictions of fault inflow in highly iterative processes.

The results show that the *concave models* performs best for nearly all predictions in all projects. Analogous to the overall performance, the *S-shaped models* perform worse compared to the concave models.

The *linear model* shows a similar performance as the concave models in the short-term prediction in $P1$ and $P2$. However, in $P3$, the linear performs worst, while still being accurate. Thus, the linear model provides a valid alternative for the short-term predictions as well.

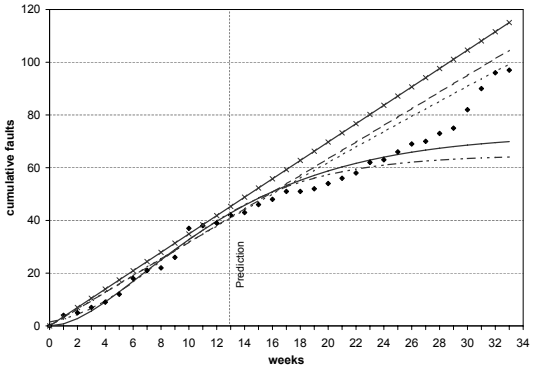
As shown in Tables 2 and 3, the short-term predictions of the SRGMs are considerably better than their overall predictions. The reason for this is, in our opinion, that the models are built on historical data, i.e. the models get adjusted to the actual fault inflow. Consequently, the models' predictions are reasonably close to the actual inflow in the next interval of measurement. However, predictions for two weeks in advance are in general less accurate than the prediction for one week, and the predictions for three weeks in advance less accurate than those for two weeks in advance respectively.

This finding is, by large, in line with the results presented in [16]. However, the predictions in our study are more accurate. Besides, our results indicate that SRGMs are suitable for predicting fault inflow as well. Moreover, the models used in our study do not require any project-related metrics.

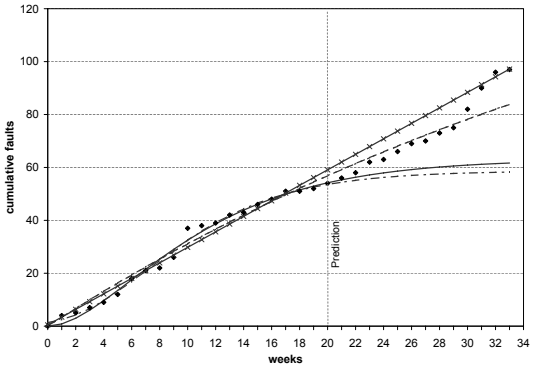
5.3. Validity

With regard to our study, different types of validity have to be taken into account [19]:

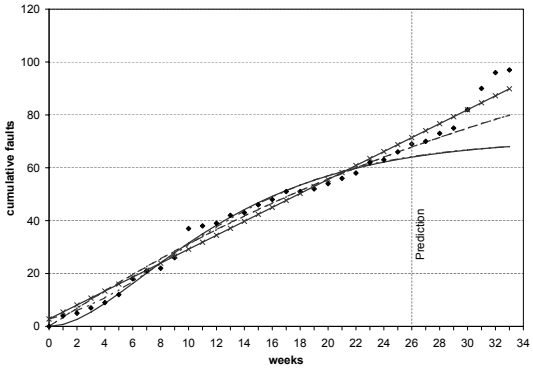
External validity refers to the ability to generalize the results. Critical aspects, in this regard, are the selection of the candidate models and the study's context. A huge variety of SRGMs has been proposed in literature. Since this study accounts for only four SRGMs, it cannot be claimed that the results are valid for all existing



(a) Project 3—Prediction at 40% project time.



(b) Project 3—Prediction at 60% project time.



(c) Project 3—Prediction at 80% project time.

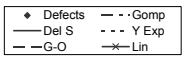


Figure 3. Project 3—Predictions at 40%, 60% and 80% of project time

SRGMs. The three projects under study were carried out by one organization and thereby, following similar development methods. However, the projects relate to different products which indicates that the study's results are valid in different contexts.

Construct validity describes the researchers' ability to measure what they are interested in measuring [19]. This relates to the way the candidate models are built and how their performance is measured. The estimation of the models' parameters was conducted using NCSS and the paired t-test was performed using Excel. Moreover, the data was tested for normality using Datalab. Different applications might produce slightly different results due to the approach used in the calculation. However, we believe that this effect does not influence the results significantly.

In order to compare the models, we considered using the χ^2 goodness-of-fit test [19] as a statistic for measuring the models' performance. This test was, however, not applicable on the available data, as the frequencies, i.e. the difference between predicted and actual values should be greater than 5 [19]. Instead, the models are compared based on the MMRE, which is widely used as an indicator for the accuracy of predictions [10]. However, some authors (e.g. [5]) claim, that the MMRE is unreliable for comparing prediction models. Therefore, further evaluation of the models based on different criteria is suggested. However, for industrial acceptance of our results we think it is an advantage to use a simple, and easy to understand, performance evaluation measure.

Conclusion validity relates to the correctness of the conclusions. Here, correctness means that the results are statistically significant. In order to increase the conclusion validity, the paired t-test was conducted as described in Section 4.2. For the short-term prediction, a statistical test could not be applied, as the the data sample for the short-term prediction is very small and thus, statistical tests would not be powerful [19].

6. Conclusions and further work

The goal of this study was to identify potential models for predicting fault inflow in highly iterative processes. Predicting fault inflow enables management to plan for required resources, i.e. man-hours in advance. Thus, resource shortages could be prevented. Moreover, management could introduce process improvements earlier in the project when the expected fault inflow is higher than desired. Consequently, the prediction of fault inflow provides for more accurate cost estimation and control.

In our study, we suggested using SRGMs to predict fault inflow. Besides, this study presented a linear model. The models were evaluated based on their ability to predict fault inflow for the remaining project time. Additionally, it was investigated how the models performed regarding short-term predictions.

First, this study evaluated the overall performance of the models. The results show, that SRGMs can be used to predict fault inflow. Moreover, it was shown that the concave SRGMs perform significantly better than S-shaped models. Additionally, it was found that a linear model provides a valid alternative to the concave SRGMs. We evaluated the models' performance at 40%, 60% and 80% of the project time. At 40% of the project time, the models showed varying results and none of the models were found to be suitable for all projects in this study. At 60% and 80% respectively, the S-shaped models perform worst, whereas the linear model and the concave models perform better. The concave models generally proved to be most stable.

Second, the results indicate, that SRGMs can be used to make short-term predictions in iterative processes. The explanation is that the models are built on fault data and, consequently, perform well in the predictions for one, two and three weeks in advance. However, the predictions become less accurate the further in advance the predictions are made. Here, the concave models outperform the S-shaped models and the linear model provides a valid alternative.

Thus, we recommend to use concave models and the linear model, respectively, to predict fault inflow. We believe, that the linear model is especially simple to use and cost-effective. However, a dynamic validation of the models is recommended, i.e. the models have to be evaluated within running projects. In that way, the models' usability for predicting fault inflow can be investigated in more detail.

Moreover, further work could be targeted towards automation. In particular, this concerns the models' parameter estimation, extracting fault data and the actual application of the models. The only condition here is that data on detected faults is already available, e.g. in a database. Besides, further work is suggested to evaluate SRGMs and the linear model to predict the total number of faults in iterative software development processes. Based on our results, the models of this study seem to be suitable for this prediction as well.

Acknowledgment

The authors would like to thank Ericsson AB in Karlskrona for supporting this study. Especially, we are deeply grateful to the persons directly contributing to this study through discussions and suggestions.

References

[1] *IEEE standard dictionary of measures to produce reliable software, IEEE Std 982.1-1988*, 1989.

[2] C. Andersson. A replicated empirical study of a selection method for software reliability growth models. *Empirical Software Engineering*, 12(2):161–182, 2007.

[3] D. Cohen, M. Lindvall, and P. Costa. An introduction to agile methods. *Advances in Computers*, 62:2–67, 2004.

[4] N. Fenton, M. Neil, W. Marsh, P. Hearty, D. Marquez, P. Krause, and R. Mishra. Predicting software defects in varying development lifecycles using bayesian nets. *Information and Software Technology*, 49(1):32–43, 2007.

[5] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion MMRE. *IEEE Transactions on Software Engineering*, 29(11):985–995, 2003.

[6] A. L. Goel. Software reliability models: assumptions, limitations, and applicability. *IEEE Transactions on Software Engineering*, 11(12):1411–1423, 1985.

[7] A. L. Goel and K. Okumoto. A Markovian model for reliability and other performance measures of software systems. In *Proceedings of the National Computer Conference*, volume 48, pages 769–774, New York, 1979.

[8] A. L. Goel and K. Okumoto. When to stop testing and start using software? In *Proceedings of the 1981 ACM Workshop/Symposium on Measurement and Evaluation of Software Quality*, pages 131–138, New York, NY, USA, 1981. ACM.

[9] D. Kececioglu. *Reliability engineering handbook (vol. 2)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.

[10] B. Kitchenham, L. Pickard, S. G. MacDonell, and M. J. Shepperd. What accuracy statistics really measure. *IEE Proceedings - Software*, 148(3):81–85, 2001.

[11] A. M. Mood, F. A. Graybill, and D. C. Boes. *Introduction to the theory of statistics*. McGraw-Hill Series in Probability and Statistics. McGraw-Hill, 3rd edition, 1974.

[12] J. D. Musa. *Software reliability engineering: more reliable software faster and cheaper*. AuthorHouse, Inc., 2nd edition, 2004.

[13] N. Nagappan and T. Ball. Use of relative code churn measures to predict system defect density. In *ICSE '05: Proceedings of the 27th International Conference on Software Engineering*, pages 284–292, New York, NY, USA, 2005. ACM.

[14] N. Ohlsson and H. Alberg. Predicting fault-prone software modules in telephone switches. *IEEE Transactions on Software Engineering*, 22(12):886–894, 1996.

[15] G. J. Pai and J. B. Dugan. Empirical analysis of software fault content and fault proneness using Bayesian methods. *IEEE Transactions on Software Engineering*, 33(10):675–686, 2007.

[16] M. Staron and W. Meding. Predicting weekly defect inflow in large software projects based on project planning and test status. *Information and Software Technology*, 50(7–8):782–796, 2008.

[17] C. Stringfellow and A. A. Andrews. An empirical method for selecting software reliability growth models. *Empirical Software Engineering*, 7(4):319–343, 2002.

[18] C. Wohlin and P. Runeson. Defect content estimations from review data. In *ICSE '98: Proceedings of the 20th International Conference on Software Engineering*, pages 400–409, Washington, DC, USA, 1998. IEEE Computer Society.

[19] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[20] A. Wood. Predicting software reliability. *Computer*, 29(11):69–77, 1996.

[21] A. Wood. Software reliability growth models: Assumptions vs. reality. In *ISSRE '97: Proceedings of the Eighth International Symposium on Software Reliability Engineering*, page 136, Washington, DC, USA, 1997. IEEE Computer Society.

[22] S. Yamada, M. Ohba, and S. Osaki. S-shaped reliability growth modeling for software error detection. *IEEE Transactions on Reliability*, 32(5):475–478, December 1983.

[23] S. Yamada, H. Ohtera, and H. Narihisa. Software reliability growth models with testing effort. *IEEE Transactions on Reliability*, 35(1):19–23, 1986.