Predicting Short-Term Defect Inflow in Large Software Projects – An Initial Evaluation

Mirosław Staron¹⁾, Wilhelm Meding²⁾

¹⁾ IT University of Göteborg
Gothenburg, Sweden
miroslaw.staron@ituniv.se

²⁾ Ericsson SW Research
Ericsson AB
Gothenburg, Sweden
wilhelm.meding@ericsson.com

Predicting a defect inflow is important for project planning and monitoring purposes. For project planning purposes and for quality management purposes, an important measure is the trend of defect inflow in the project – i.e. how many defects are reported in a particular stage of the project. Predicting the defect inflow provides a mechanism of early notification whether the project is going to meet the set goals or not. In this paper we present and evaluate a method for predicting defect inflow for large software projects: a method for short-term predictions for up to three weeks in advance on a weekly basis. The contribution of this paper is the fact that our model is based on the data from project planning, status monitoring, and current trends of defect inflow and produces results applicable for large projects. The method is evaluated by comparing it to existing defect inflow prediction practices (e.g. expert estimations) at one of the large projects at Ericsson. The results show that the method provides more accurate predictions (in most cases) while decreasing the time required for constructing the predictions using current practices in the company.

Keywords: Prediction models, defect inflow, project status monitoring.

1. INTRODUCTION

The number of defects in a software project has a significant impact on project performance and hence is an input to project planning. As the quality level of the final product is set at the beginning of the project, a large number of defects can result in project delays and cost overruns. For project planning purposes and for quality management, an important measure is the trend of *defect inflow* in the project – i.e. how many defects are reported in a particular time. As opposed to metrics related to defect density which are calculated for each component, the defect inflow is a dynamic metric and changes over time. The defect inflow is a measure which is eminent on the project level and depends on the sub-projects (or work package – c.f. Section 3) testing phase. Defect inflow is one of the most important variables to monitor in large scale software projects. It provides the management with a possibility of identifying whether a given project is not going to meet the set goals and to adjust the project plan, if needed. It allows also the organization to optimize resource allocation for projects – e.g. when there is a large defect inflow, the organization needs to provide additional person-hours to keep the project on track (e.g. by ordering overtime). It is used in line with effort estimations methods (e.g. COCOMO/COCOMO II (Boehm, 2000)).

Large software projects have very different dynamics to small projects; the number of factors that affect the project is much larger than for small and medium software projects. Large software projects also tend to develop complex software-hardware systems. Therefore, while constructing predictions or estimations for large software projects, there is a trade-off between the prediction accuracy and the effort required to collect the data required to predict. In large projects it is of particular importance as the data might be distributed over time and across the globe. The current practices for large software projects at Ericsson rely heavily on expert estimations, which are rather time consuming; in particular the experts use Case Based Reasoning (CBR, (Maiden and Sutcliffe, 1993)) while constructing the predictions for defect inflow – by identifying similarities and differences between projects the experts construct the predictions. The goal of introducing a new method in this paper is to provide the experts with support for creating the prediction models using statistical methods based on the data which is already collected in the organization (or which can be collected at reasonable cost). This approach resulted in a method which is simple and which has high-cost efficiency (e.g. the costs of mis-predictions are smaller to the costs of building and maintaining more accurate models), which could be seen as a trade-off between prediction accuracy and costs of predicting. In this paper we try to address this trade-off by minimizing the number of measurements to be collected while focusing on existing established measurements in the organization while at the same time minimizing the cost for data reconfiguration. However, in the course of our research new ways of data collection were introduced, which improved the practice and allowed for more accurate predictions.

In this paper we address the following research questions:

RQ1: How can we predict a short-term defect inflow, based on the analysis of current defect inflow and project plans?

RQ2: How accurate are the predictions made using the new method compared to alternative methods?

These research questions were posed in the context of the development of a complete measurement system for large software projects at Ericsson. The intention of our prediction models was to build on the existing measurements by introducing more advanced usage scenarios for these measurements. As a result of addressing the first question we developed a short-term prediction model using multivariate linear regression based on project plans and the current defect inflow. We also validated this model for other measurements – e.g. defect fix time prediction; the results of that evaluation, however, are not in the scope of this paper. The evaluation of the new method in a project at Ericsson provided us with the possibility to address the second research question. An important aspect in this paper is the context of the projects at Ericsson. This research is done in projects which are structured around work packages rather than subsystems or subprojects and our contribution in this paper is two-fold:

1. we use project metrics (e.g. milestone completion status for work packages) in the prediction models and not product metrics (e.g. number of lines of code changed),
2. we predict defect inflow in the development project, and not after the release of the product to the market.

This paper is structured as follows. Section 2 presents the most related work in the field and discusses the differences of our approach (justifying our decisions on choosing the particular approach) and the alternative approaches. Section 3 introduces the design of our case study at Ericsson, including the context of the company, methods used for developing prediction models, and methods for evaluating the prediction models. Section 4 presents the short-term prediction models and Section 5 provides the evaluation of the models. The conclusions and further work are presented in Section 6.

2. RELATED WORK

In our work we consider the defect inflow to be the function of characteristics of work packages (e.g. the accumulated number of components reaching a particular milestone) and not directly the characteristics of the affected components (e.g. size or complexity). Using the characteristics of components as the sole predictors would provide us with a possibility to predict the defect density of the component and present this data on a monthly/weekly basis (based on when the component will be put under testing). Such an approach would be an extension of the current work on defect density, e.g. (Ball and Nagappan, 2005a, Neufelder, 2000, Malaiya and Denton, 2000, Mohagheghi et al., 2004, Agresti and Evancho, 1992, Ball and Nagappan, 2005b). In our case, nevertheless, this approach seems not feasible, because the information about how the components are to be affected by the project is not available at the time of developing predictions; in particular the change of size and complexity is not available. Furthermore, for large software projects, the predictions of defect densities have been found to be insufficient (Fenton and Neil, 1999). For short-term predictions, the data on size and complexity of components was not available on a weekly basis simply because measuring the size and complexity change is not meaningful for particular weeks; the measurements of component characteristics are done according to project plans – e.g. builds – and not on a weekly basis (i.e. not according to calendar time). In our further work we intend to evaluate whether it is feasible to re-configure this data and use it as an auxiliary prediction method.

Predicting the defect inflow has been researched previously and general purpose models, such as the Rayleigh model (Laird and Brennan, 2006), have been introduced based on the reliability theory (Fenton and Pfleeger, 1996). As these models are very general, they are bound to have prediction errors due their generality. However, we have found in the course of our study that the defect inflow distribution does not follow the Rayleigh (or Weibull) distribution, which renders the Rayleigh model only partially useful.

Another alternative approach is to use data on test cases to predict the number of defects (e.g. using such methods as Capture-Recapture (Isoda, 1998, Thelin and Runeson, 1999). For the short-term predictions we intend to use this in the next step of our research, however, this data still needs reconfiguration since it suffers from the same problems as mentioned at the beginning of this section – i.e. being done on a subsystem/component basis and not on the work package basis.

Although the prediction of defect inflow seems to be closely related to the area of reliability modeling, it differs significantly from it. In particular, the reliability modeling is concerned about the software reliability after release, e.g. (Cavano, 1984, Littlewood et al., 1991, Bailey and Kowalski, 1992), while our research is focused on the rate of defect inflow (i.e. discovery and reporting) during the project development. Our intention is to extend this work into predicting the post-release reliability based on pre-release defect inflow.

Finally, our research is related to the research on fault slip-through (Damm and Lundberg, 2005). In our research we intend to use the methods for predicting fault slip-through as the next refinement of this method. However, at this point of time the data for fault slip-through was not available at weekly basis due to the definition of this measurement. We intend to work on using this measurement to improve this work.

3. CASE STUDY DESIGN

The methods for defect inflow prediction presented in this paper were developed in an empirical way in close collaboration with Ericsson and in particular with the quality management group. The case study provided us with the possibility to both develop the defect inflow prediction methods, and to evaluate them in a large scale software projects. This section presents the design of both of these parts. In general, the goal of our study could be characterized in the following way, as advocated by (Mashiko and Basili, 1997):

The goal of this study is to increase the efficiency of defect inflow prediction in large software projects from the perspective of a quality manager.

In this goal we perceive the increase of efficiency as the increase of accuracy and, if possible, a decrease in time required to make the predictions.

3.1. Context

The context of the case study is Ericsson AB and one of its large projects, which is the development of one of the releases of a network component. It is an embedded, distributed, and mission critical software, which is developed using the Rational Unified Process. The size of the project varies between 100 and 200 persons depending on the phase of the project. The processes in the project are stable and mature. In the course of development of the methods we used one of the previous releases of the product (the previous one – release 7) and we applied the methods for the new release of the product (release 8). Choosing late project releases decreases the risk of using data biased with immaturity in the organization. This has already been shown in (Tomaszewski and Lundberg, 2005, Tomaszewski and Lundberg, 2006) in a similar context at the same company.

As a new practice at Ericsson, the large software projects are structured into a set of work packages which are defined during the project planning phase. The number of defects manifested at a particular point in time seems to be a function of which of the work packages reach the testing phase, and not so much about the components which are affected. At the project management level, the defect inflow measure is directly related to work packages, and indirectly related to particular components which are affected by the work packages. In the existing prediction work on defect density (Ball and Nagappan, 2005a, Neufelder, 2000, Malaiya and Denton, 2000, Mohagheghi et al., 2004) it is usually the case that a component is developed by a single work package (or even the project, depending on the size of the component and project). However, in the case of Ericsson, the work packages are related to the functionality being developed and not to the components affected. The relationship between work packages and system components is presented in Figure 1.

Figure 1. Relationship between work packages and components in large software projects at Ericsson

In Figure 1, the large software project is divided into smaller work packages which affect the particular components. The division of project into work packages is based on customer requirements, while the division of system into sub-systems and components is based on such elements as architectural design and the architecture of the underlying hardware (hardware/environment constraints). Each work package develops (or makes changes to) different components for each new large project, which makes it hard to develop a unified defect inflow prediction model using measurements at the component level. During the whole product life cycle (which spans over more than one large project – also referred to as *release* from the perspective of the product) the division of projects into work packages changes to a large extent (as the requirements are different for every release). Therefore using only work package characteristics (which are based on distributing functionality) makes the method for short-term predictions generalizable to other projects at Ericsson.

3.2. Development of the method

Our method for constructing the short-term prediction models is to use historical data from the defect inflow trends and project plans and the method results in developing prediction models. A short-term prediction model is a linear regression model over the data from project plan and the current defect inflow prediction. The rationale behind this regression model is that the defect inflow can be predicted from the data available for the current project. We constructed our prediction models based on the following assumption: we use data from past weeks to describe the defect inflow for a given week. This assumption means that once we have the regression model describing the defect inflow for a particular week using data from the past weeks, we can use this model to predict the defect inflow for the coming weeks when we substitute the data for past weeks with the data from the current week. As the predictor variables we use the defect inflow for the current project (for the time up to till the current week), and milestone completion status. The choice of the candidate predictors was the availability of data and our goal – to make predictions based on the data that already existed in the organization and was easy to obtain. We distinguish between two milestones in the project – design and implementation ready project milestone (DRM) and test ready milestone (TRM). Testing is performed before both of these milestones, which is one of the sources of defect inflow. DRM can be taken when all work packages complete their design (we use the work package milestone Md) and TRM can be taken when all work packages complete their testing (all work packages complete the work package milestone Mt). Md status is only used before the DRM date for the project (after that date, the accumulated number of Md completion is constant and the number of Md completion per week is zero). After the DRM date the Mt completion status is used.

While working with measurements at Ericsson, we identified the following factors at the project level, as potentially influencing the defect inflow for a given week:

- The number of work packages that complete design and/or testing during a given week and during previous weeks (as defects might be reported with some delay)
- The number of work packages that are planned to complete design and/or testing during a given week and during previous weeks
- The number of defects that were reported up to 5 weeks before a given week, as we identified that high number of defects in a given week is related to the number of defects in previous weeks

The variables in Table 1 measure these factors and interviews with experts at the company supported our assumptions about the influences of these variables on the defect inflow. We chose the above measurements from the set of few hundreds measurements being collected in the company as we perceived the above factors as having a key influence on the number of defects.

Measurement	Prefix	Description	Variables used	Example
Number of planned Md completions	Mdp	The number of work packages planned to complete Md in the given week	Mdp5, Mdp4, Mdp3, Mdp2, Mdp1, Mdp0	Mdp0 = 2: for the predicted week, 2 work packages were planned to complete Md
Number of actual Md completions	Mda	The number of work packages actually completing Md in the given week	Mda5, Mda4, Mda3, Mda2, Mda1 ¹	Mda1=1: one week before the predicted week, 1 work package completed Md
Number of planned Mt completions	Mtp	The number of work packages planned to complete Mt in the given week	Mtp5, Mtp4, Mtp3, Mtp2, Mtp1, Mtp0	Mtp4 = 3: four weeks before the predicted week, 3 work packages were planned to complete Mt
Number of actual Mt completions	Mta	The number of work packages actually completing Mt in the given week	Mta5, Mta4, Mta3, Mta2, Mta1	Mta2=1: two weeks before the predicted week, one work package completed Mt
Number of reported defects (defect inflow)	Di	The number of reported defects	Di5, Di4, Di3, Di2, Di1	Di1=10: one week before the predicted week, there were 10 defects reported
Accumulated number of planned Md completions	AMdp	The number of work packages planned to complete Md from the beginning of the project up to the given week	AMdp5, AMdp4, AMdp3, AMdp2, AMdp1, AMdp0	AMdp0 = 20: up until the predicted week, 20 work packages were planned to complete Md
Accumulated number of actual Md completions	AMda	The number of work packages actually completing Md from the beginning of the project up until the given week	AMda5, AMda4, AMda3, AMda2, AMda1	AMda1=18: up until one week before the predicted week, 18 work packages completed Md
Accumulated number of planned Mt completions	AMtp	The number of work packages planned to complete Mt from the beginning of the project up until the given week	AMtp5, AMtp4, AMtp3, AMtp2, AMtp1, AMtp0	AMtp4 = 8: up until four weeks before the predicted week, 8 work packages were planned to complete Mt
Accumulated number of actual Mt completions	AMta	The number of work packages actually completing Mt from the beginning of the project up until the given week	AMta5, AMta4, AMta3, AMta2, AMta1	AMta2=12: up until two weeks before the predicted week, 12 work packages completed Mt

Table 1. Measurements used for building prediction models – all measurements are per week

¹ The actual number of work packages completing Md in the predicted week is not known by the time the predictions are made. The same is valid for all measurements of *actual* values.

The rationale behind the above candidate variables is that our goal was to prepare a short-term defect inflow prediction model based on the data from the current project. By using the data from 1–5 weeks before, we could construct a regression model describing the defect inflow using historical data for 1–5 weeks. To construct the model we used the previous release of the product, which we found to be similar in size, complexity, and project team maturity to the projects for which the model was to apply. Although it would be possible to construct the model from the data for the new release (8th), our goal was to build a model from the data from a completed project. This allows us to predict problematic weeks in the new project (if the same combination of factors as in the previous project takes place). The constructed model will allow predicting the defect inflow for 1-5 weeks in advance, for example if we find that our predictor variables are the number of defect inflow from 5 weeks before and the accumulated planned Md completion, using the data for the current week, we could predict what the defect inflow will be in 5 weeks. However, we have found by doing simulations that the predictions for 4 and 5 weeks in advance have low prediction accuracy (for the data the models were constructed from) and therefore they are not discussed in this paper.

In addition to using the data from the project plan (Md/Mt completion status for work packages) we could use the data for planned and executed test cases. However, the accumulated numbers of test cases planned and executed were highly correlated with the Md/Mt completion status (Spearman's correlation coefficients between 0.96 and 0.99 significant at the 0.01 level). To avoid problems with multi-collinearity we used the variables that were not correlated and we chose the data from the project plan which could be obtained earlier in the project. A representative scatter plot for relationship between two of these variables is presented in Figure 2 (Spearman's correlation coefficient: 0.96).

Figure 2. Relationship between accumulated planned test cases to execute and Accumulated Md closures

We deliberately do not include the data for product size/complexity as this data was not related to project planning for the following reasons:

- the software components were not assigned on a one-to-one basis to work packages – and the milestones characterized the work packages, not components
- the data on source code size was collected for milestones in the project (as it does not make sense to collect them on a weekly basis) – which means that for the whole project we could use few data points for size
- the organization was concerned with project planning and monitoring, and not source code characteristics (e.g. size is only an input to the planning, but is not monitored in the projects)

Before constructing the regression model over the candidate variables, Principal Component Analysis (PCA) is used to reduce the number of variables thus identifying the strongest predictors. For each prediction model we used a modified set of variables – e.g. for building the model for 3 weeks we did not use some data, such as the actual number of work packages completing Md 2 weeks in advance. This is because while predicting the defect inflow in the future project, when we predict 3 weeks in advance, we do not have the actual number of work packages completing Md for two weeks in advance.

The variables which had the strongest loadings in the strongest principal component were used as input to stepwise linear regression which resulted in constructing the prediction model. After the model is constructed we use the model fit coefficient (R^2) for initial assessment of accuracy of the model. We also use the scatter plots to evaluate visually whether the relationship between the principal component and defect inflow is linear.

3.3. Evaluation of the method

The short-term prediction method is evaluated before being deployed into projects. In this case study we used the historical data for the previous releases of the project (release 6 and 7 respectively). We also use comparison of

the new predictions with the predictions created through expert opinions and Case-Based Reasoning (although in a limited manner).

The predictions are constructed using multivariate linear regression methods. As part of the result of regression methods, we obtain the model fit coefficient (R^2), which we use as the indication of appropriateness of the model. This coefficient, however, does not allow evaluating how well the model predicts the actual data from new projects. Therefore we also evaluate the predictions on historical data from existing projects at Ericsson (in particular we do not use the projects which were used to construct the models). We calculate the Mean Square Error (MSE) for the predictions to compare various short-term models:

$$MSE = \frac{\sum_{i=1}^n (a_i - p_i)^2}{n}$$

In the formula, a_i denotes the actual defect inflow value for the i -th observation, p_i denotes the predicted defect inflow value for the i -th observation, and n denotes the number of observations (in our case each observation is a week). The best models are expected to have the lowest value of MSE – i.e. the mean square error of estimations is small.

In addition to using MSE we also show the distribution of the absolute relative error (ARE) which is defined as:

$$ARE = ABS\left(\frac{a_i - p_i}{a_i}\right)$$

The absolute relative error shows how much the predictions differ from the actual data. Instead of providing a single value, the average, we provide a distribution of the absolute relative error as it provides a better picture over the accuracy of the predictions.

3.3.1. Reference prediction models for short-term predictions

In evaluation of the prediction accuracy we compare models developed in our research (presented in Section 4). We also evaluate that against “average” models – i.e. predicting using a simple average amount of defect inflow in a baseline project (or the average number of defects in the current project – up to the week for which the prediction is made). The rationale behind the average models is that if we do not know how to predict the number of defect inflow in a particular week, we could take the average number of defects for all weeks as an indicator; alternatively we can also use the median (i.e. the most common value of the defect inflow). For comparison we use the following models:

- Average number of defect inflow from the baseline project (to some extent this is the use of Case Based Reasoning)
- Average number of defect inflow from the actual project so far
- Moving average (2 weeks) of the number of defect inflow from the current project (i.e. the predicted value of the defect inflow is the average of the defect inflow from previous 2 weeks)
- Moving average (3 weeks) of the number of defect inflow from the current project (i.e. the predicted value of the defect inflow is the average of the defect inflow from previous 3 weeks)
- Value of the mode of the defect inflow from the baseline project (to some extent this is the use of Case Based Reasoning)
- Value of the mode of the defect inflow from the current project
- Expert estimations for 1 week
- Expert estimations for 2 weeks
- Expert estimations for 3 weeks

For each of these models we calculate MSE and ARE. The best model is expected to have the lowest value of the MSE and it should have a distribution of ARE centered around the mean (i.e. having small variability of mis-predictions).

3.4. Threats to validity of the study

The threats to validity are grouped into four groups as recommended by Wohlin et al. (2000).

One of the main external validity threats for our research is the generalizability of the results and the methods. One of the main threats is the suitability of the methods for other projects than the ones we worked with. We apply the same methods in other projects at Ericsson. It seems that the methods work well for other projects as long as the projects are structured along work packages and not sub-projects. This makes our method more suitable for

projects done according to the principles of agility – i.e. shorter releases and focusing on work packages/functionality rather than sub-projects.

One of the main construct validity threats in our study is the choice of metrics used to construct the models, in particular focusing only on measurements related to work packages and not structural characteristics of the underlying source code. Although being a threat, it was also a goal of our study to evaluate the suitability of such measurements for predicting defect inflow. Collecting more advanced data would result in increases costs of predictions, which would overweight the costs of mis-predictions. We also wanted to provide empirical evidence on how suitable our identified metrics are in this context. Through experiments with the complete data set (over 100 measurements) and interviews with experts we found that the measurements we use in our study strongly influence the number of defects reported. We deliberately do not consider software components in our study as the required information is not suitable for our analyses – it is not known in which week a defect will be found in particular software component.

One of the conclusion validity threats is the fact that we did not remove outliers from our data set. This was a conscious decision as the outliers are data points which are the most interesting ones to predict. A very high defect inflow rate is dangerous for projects and therefore constitutes information needed by the project management. Removing the statistical outliers would decrease the value of our prediction models for the company.

We constructed our study to minimize the threats to the internal validity and therefore there are no threats for this category.

4. SHORT-TERM DEFECT INFLOW PREDICTION MODEL

Using the Principal Component Analysis we reduced the data set to key variables. We experimented with the initial set of variables (the input to PCA) in order to achieve the best possible percentage of explaining the variability at the minimal set of measurements. The scatter plot for the main principal components and the defect inflow is presented in Figure 3. Due to the confidentiality of the data the values on the Y-axis are not provided.

Figure 3. Scatter plot for the main principal components and defect inflow for the models before DRM and after DRM. The X-axes show the values of the components

The principal components before DRM seem to be linearly correlated to the defect inflow, which makes the linear regression a viable technique for building the prediction model in this case. For the principal components after DRM, the components do not show a strong correlation with the defect inflow, which affects the fitness of the regression models and the prediction accuracy. We have also checked whether the principal components expose logarithmic and polynomial relationship to the defect inflow, but the results showed almost a complete lack of relationship for the logarithmic (the points grouped along a horizontal line) and large scatter for the polynomial relationship. The variability of the data set explained by the principal components is presented in the last column in Table 2.

The equations used to predict the short-term defect inflow are presented together with the R^2 coefficient for the regression model. The variables used in the equations are subsets of variables presented in Section 3. We have used PCA to identify the key components and we used variables which constituted these components in building the prediction models. We build the models using principal components, but we have re-calculated the loadings in the components so that we could present the equations using the original variables and not the components (as this was one of our requirements while deploying the model in the company – to use the original names of measurements, not the name of the component).

Before/after DRM (design ready milestone)	Period	Equation	R ²	Var. expl. by comp.
Before DRM	1 week	$D = 1.499 + 0.584*AMdp0 + 0.650*Di1 - 1.285*AMdp1 + 1.102*AMda1$	0.86	93.84%
	2 weeks	$D = 2.639 + 1.173*AMdp0 - 2.029*AMdp2 + 1.724*AMda2 + 0.461*Di2 - 0.366*Di3$	0.82	91.43%
	3 weeks	$D = 4.187 - 0.357*Di3 + 1.928*AMdp5 - 1.192*AMdp0$	0.62	90.92%
After DRM	1 week	$D = 39.155 + 0.470*Di1 + 1.290*AMtp0 - 1.185*AMtp1 - 1.214*AMta1 + 0.039*AMtp2 - 0.044*AMta2 + 1.297*AMtp3 - 0.517*AMta3 + 0.003*AMtp4 + 0.107*AMta4 + 0.148*Di2 - 0.237*Di3 - 0.194*Di4 + 0.180*Di5$	0.67	65.74%
	2 weeks	$D = 53.669 + 1.419*AMtp0 - 0.682*AMtp1 + 0.099*AMtp2 - 1.844*AMta2 + 0.429*AMtp3 - 0.768*AMta3 + 0.646*AMtp4 + 0.446*AMta4 + 0.306*Di2 - 0.265*Di3$	0.55	78.69%
	3 weeks	$D = 24.82 + 0.47*Di3 - 0.351*AMtp2 + 0.308*Di5$	0.62	59.09%

Table 2. Defect inflow prediction models (short-term)

As an example, let us predict the defect inflow for a particular week in an example project². Let us assume that we want to predict week 13 in the project, which is before the design ready milestone. The data for that particular week is presented in Table 3 in the shaded area. For the predictions for week 13, we use the coefficients from the equations in Table 2 (Period = 3 weeks, before DRM); they are presented in the bottom row in Table 3 (C-3w – coefficients for 3 weeks). For the predictions for week 12, we use another equation from Table 2 (Period = 2 weeks, before DRM) and we need to use the variables in a different way – AMdp0 from the original equation (Table 2) becomes AMdp1 when used in Table 3 because week 0 in the case of 2 weeks prediction is week 12, not 13; we need to change the coefficients in the same way for all variables for 1 week and 2 weeks prediction.

	AMdp0 – week 13	AMdp1 – week 12	AMdp2 – week 11	AMdp3 – week 10	AMdp5 – week 8	AMda3 – week 10	Di3 – week 10	Di4 – week 9	Consta nt	Predict ed value
	19	18	18	17	16	11	5	10		
C-1w			0.584	-1.285		1.102	0.65		1.499	22
C-2w		1.173		-2.029		1.724	0.461	-0.366	2.639	22
C-3w	-1.192				1.928		-0.357		4.187	11

Table 3. Data used for predicting week 13, 12, and 11 in the project

The results for the short-term predictions are presented in Figure 4.

Figure 4. Short-term defect inflow prediction for week 10

The figure shows that given the current circumstances of the project (i.e. the number of defects reported in the current week and the status of the planned and accumulated numbers of work packages reaching the DRM milestone) there are will be a short-term (2 weeks) raise in the defect inflow in the project, but it will drop after 3 weeks.

5. EVALUATION OF THE METHOD

In Section 4, the prediction models were initially evaluated from the perspective of how well they describe the baseline project which was used to construct the model. In this section we evaluate the models for another project.

² Due to the sensitivity of the project data at Ericsson, the actual project data cannot be used in this example.

The evaluation project is the next (8th) release of the same product, while the baseline project was the previous (7th) release of the same product.

The values for the MSE for the reference prediction models and the short-term prediction models are presented in Figure 5. The figure indicates that the best model is our model for 1 week prediction and for 2 week prediction. Relatively accurate models are moving averages, but their main disadvantage is that they are not predictions, but descriptions of existing trends in defect inflow.

Figure 5. MSE for short-term predictions

The next step in the evaluation is to provide a distribution of the absolute relative error for all the predictions. The results are graphically presented in Figure 6.

Figure 6. Box-plots for distributions of absolute relative errors of predictions (presented in two different charts since the number of data points for experts was considerably lower than for other methods)

From the experiments with the historical data we found that the prediction models developed in this paper had a tendency of over-predicting (i.e. predicting values that were larger than the actual values), in particular indicating the potential “red-alerts” for the projects – i.e. showing that there will be a high raise in the defect inflow in the project. Although this might be a problem from the statistical perspective (low accuracy), this provides a means for project managers to get early warnings of potential problems so that they can have a time for reacting to some extent. This, however, cannot be verified on the historical data and we are currently in the process of evaluating the models in other large projects at Ericsson.

6. CONCLUSIONS AND FURTHER WORK

In this paper we presented the method for providing short term predictions of defect inflow in large software projects based on the data from project plans and the defect inflow itself from the past weeks of the projects. We compared the developed regression models with other ways of constructing the predictions – e.g. using averages or expert estimations. Our goal in this study was to increase the efficiency of predicting the defect inflow by increasing the accuracy and decreasing the time required to construct the predictions (this is achieved by automation of the process of calculating the predicted values).

The results of our research show that the accuracy of the models is dependent on the data that is the input for the predictions. The multivariate linear regression models were found to produce models which provided us with the largest percentage of accurate predictions, while they also sometimes mis-predicted the defect inflow to a large extent. It was also found that for the large software projects structured around work packages (and not sub-projects), project planning is accurate enough to be used for constructing short term predictions. The project planning based on work packages accurately accounts for organizational issues such as holidays. The deviations from the planned closing of work packages also has an influence on the number of defects reported in the project and, together with the accumulated numbers of planned closings of work packages and defect inflows from previous weeks, provide means of predicting the defect inflow. It was our perception that these three measures: accumulated number of planned work package closings, accumulated number of actual work package closing, and the defect inflow, provide an accurate overview of the overall condition of the project. Using these measures is the first step to identifying problems in projects (e.g. predicting a high defect inflow rate).

Our recommendation, based on the research presented in this paper, is to use the prediction models which we develop in this paper for short-term defect inflow prediction in large software project. However, a further evaluation of the method is needed in a long-term (using the predictions in projects that are currently run).

Our further work is focused on deploying the methods to other large scale projects at Ericsson and transferring the knowledge how these models should be customized if a need arises. We are also working on the modification of this method in order to predict the defect inflow for longer periods of time. We could predict weeks 4-6 using the same method by using the short-term predictions as the surrogate values for the actual defect inflow prediction. The method, however, has not yet been empirically validated, which is the mainstream of our current work.

ACKNOWLEDGEMENTS

The authors would like to thank the Software Architecture Quality Center and Ericsson AB for support in this study. We would also like to express our deepest gratitude for the persons involved and contributing to this study at Ericsson, in particular the experts we had the opportunity to work with.

REFERENCES

- AGRESTI, W. W. & EVANCO, W. M. (1992) Projecting software defects from analyzing Ada designs. *Software Engineering, IEEE Transactions on*, 18, 988-997.
- BAILEY, J. H. & KOWALSKI, R. A. (1992) Reliability-growth analysis for an Ada-coding process.
- BALL, T. & NAGAPPAN, N. (2005a) Static analysis tools as early indicators of pre-release defect density.
- BALL, T. & NAGAPPAN, N. (2005b) Use of relative code churn measures to predict system defect density.
- BOEHM, B. W. (2000) *Software cost estimation with Cocomo II*, Upper Saddle River, NJ, Prentice Hall.
- CAVANO, J. P. (1984) Software reliability measurement: Prediction, estimation, and assessment. *Journal of Systems and Software*, 4, 269-275.
- DAMM, L. O. & LUNDBERG, L. (2005) Identification of test process improvements by combining fault trigger classification and faults-slip-through measurement.
- FENTON, N. E. & NEIL, M. (1999) A critique of software defect prediction models. *Software Engineering, IEEE Transactions on*, 25, 675-689.
- FENTON, N. E. & PFLEEGER, S. L. (1996) *Software metrics : a rigorous and practical approach*, London, International Thomson Computer Press.
- ISODA, S. (1998) A criticism on the capture-and-recapture method for software reliability assurance. *Journal of Systems and Software*, 43, 3-10.
- LAIRD, L. M. & BRENNAN, M. C. (2006) *Software measurement and estimation : a practical approach*, Hoboken, N.J., John Wiley & Sons.
- LITTLEWOOD, B., FENTON, N. E. & CITY UNIVERSITY CENTRE FOR SOFTWARE RELIABILITY (1991) *Software reliability and metrics*, London, Elsevier Applied Science.
- MAIDEN, N. & SUTCLIFFE, A. (1993) Case-based reasoning in software engineering.
- MALAIYA, Y. K. & DENTON, J. (2000) Module size distribution and defect density.
- MASHIKO, Y. & BASILI, V. R. (1997) Using the GQM paradigm to investigate influential factors for software process improvement. *Journal of Systems and Software*, 36, 17-32.
- MOHAGHEGHI, P., CONRADI, R., KILLI, O. M. & SCHWARZ, H. (2004) An empirical study of software reuse vs. defect-density and stability.
- NEUFELDER, A. M. (2000) How to predict software defect density during proposal phase.
- THELIN, T. & RUNESON, P. (1999) Capture-recapture estimations for perspective-based reading-a simulated experiment. *International Conference on Product Focused Software Process Improvement*, 22-24 June 1999. Oulu, Finland, Tech. Res. Centre Finland.
- TOMASZEWSKI, P. & LUNDBERG, L. (2005) Software development productivity on a new platform: an industrial case study. *Information and Software Technology*, 47, 257-269.
- TOMASZEWSKI, P. & LUNDBERG, L. (2006) The increase of productivity over time--an industrial case study. *Information and Software Technology*, 48, 915-927.
- WOHLIN, C., RUNESON, P., HÖST, M., OHLSSON, M. C., REGNELL, B. & WESSLÉN, A. (2000) *Experimentation in Software Engineering: An Introduction*, Boston MA, Kluwer Academic Publisher.