Global Sensitivity Analysis of Predictor Models in Software Engineering

Stefan Wagner
Institut für Informatik
Technische Universität München
Boltzmannstr. 3, 85748 Garching b. München, Germany
wagnerst@in.tum.de

Abstract

Predictor models are an important tool in software projects for quality and cost control as well as management. There are various models available that can help the software engineer in decision-making. However, such models are often difficult to apply in practice because of the amount of data needed. Sensitivity analysis offers provides means to rank the input factors w.r.t. their importance and thereby reduce and optimise the measurement effort necessary. This paper presents an example application of global sensitivity analysis on a software reliability model used in practice. It describes the approach and the possibilities offered.

1. Introduction

Quality and cost management as well as other predictive tasks are still difficult to handle in the software engineering domain. This field is comparably young and the intangibility of software renders useful measurement hard. However, variety of predictor models have been proposed that aim at simplifying this problem, e.g. [1, 2, 5, 7, 12]. The models, empirical as well as analytical, allow the software engineer to predict certain aspects of a software using various input factors.

Problem. These models, however, are often difficult and/or elaborate to apply in practice. They usually involve various measurements of many input factors that influence the outcome. This amount of effort is often not possible to be spent and hence it hampers the use of the models.

Contribution. Based on the experience described in [16], we propose an application of global sensitivity analysis to predictor models in software engineering. This approach facilitates the determination of the factors that are most beneficial to determine in more detail and those that might be removed from the model. Thereby, the effort spent for applying predictor models can be optimised.

Outline. We first give a general introduction to sensitivity analysis in Sec. 2 with a special emphasis on global SA. Sec. 3 gives a brief overview of how to apply such techniques in a software engineering setting. The example of a software reliability model is described in Sec. 4. More applications of the method on predictor models are shown in Sec. 5. We then compare related work in Sec. 6 and conclude in Sec. 7.

2. Sensitivity analysis

Mathematical models, such as predictor models, are defined by equations, input factors, parameters, and variables aimed to characterise the process being investigated. The input is subject to many sources of uncertainty including errors of measurement, absence of information and poor or partial understanding of the driving forces and mechanisms. For example, costs of software defects are notoriously difficult to obtain and an exact estimation of the expected size of a system to be built is nearly impossible. This limits our confidence in the output of the models. Hence, we want to provide an evaluation of the confidence in the model, possibly assessing the uncertainties associated with the modelling process and with the outcome of the model itself. Sensitivity analysis allows to characterise the uncertainty associated with a model. This helps to answer questions like "what input factor needs to be investigated in more detail?" or "what input factors can be removed?".

2.1. Types of Analyses

Following [9] there are three main types of sensitivity analysis:

1. Screening

Third International Workshop on Predictor Models in Software Engineering (PROMISE'07)
0-7695-2954-2/07 $20.00 © 2007 IEEE

2. Local sensitivity analysis

3. Global sensitivity analysis

Screening methods are approximate but with low computational effort. This is useful when the model is expensive to compute and/or it has a huge number of input factors. Then a screening experiment can identify a subset of input factors that is most likely to have a strong effect on the model output.

Local sensitivity analysis looks at the local impact of each factor on the model output. This is usually done by computing partial derivatives of the output functions w.r.t. input factors. This approach does not make use of the further knowledge we might have of those factors. Moreover, the method does not work when the model is either nonlinear or several input factors are affected by different uncertainties. In such cases global sensitivity analysis should be used which is explained in the next section.

2.2. Global sensitivity analysis

There are several different methods that belong to the class of global sensitivity analysis. They all apportion the output uncertainty to the uncertainty of the input factors. The input factors are described by probability distribution functions that represent our knowledge of the factors. This leads to two advantages that we cite directly from Saltelli [9] who calls them the two global properties:

1. *The inclusion of influence of scale and shape*: The sensitivity estimates of individual factors incorporate the effect of the range and the shape of their probability density functions.

2. *Multidimensional averaging*: The sensitivity estimates of individual factors are evaluated varying all other factors as well.

A further important aspect is that global methods have the *model independence property*, i.e. the actual linearity or additivity of the model does not influence the functioning of the method.

Typically, global sensitivity methods belong to the class of *sample-based* methods. This means that they sample input data in order to evaluate the uncertainty in the outcome as depicted in Fig. 1. On the left-hand side, there are the input factors x_1 to x_n with their respective distributions. The dots inside those distributions represent sample points that were generated by a sampling technique, usually a Monte-Carlo method. The values for each input factor are fed to the model that calculates the corresponding result(s) Y. After repeating this for a large number of samples, we get the uncertainty distribution for the output as shown on the right-hand side. This distribution is then used to analyse the effect of the inputs.

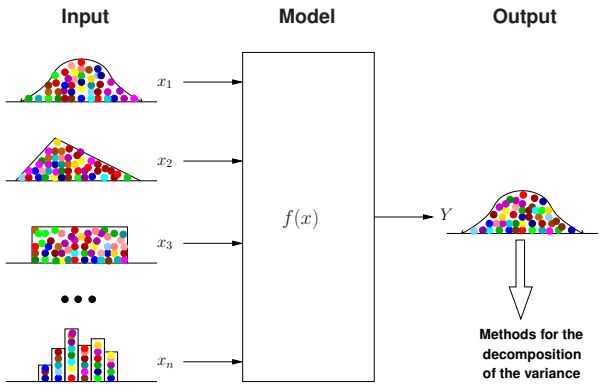

Figure 1. An overview how sampling-based methods work (adapted from [9])

For this analysis, we have the choice between several methods. A simple but effective methods is to use *scatterplots*. A scatterplot visualises the relation of an input factor to the output by plotting a large number of points with the input value on the x-axis and the output value on the y-axis. This way, possible relations become visible and possible errors can be identified. Moreover, standard correlation and regression measures such as the *Pearson product moment correlation coefficient* (PEAR) or the *Spearman coefficient* (SPEAR) can be use to identify such relationships.

Finally, there are *importance measures* that quantify the variance-decomposition of the input factors using so-called *sensitivity indices* [3]. They are divided into the *main* or *first-order indices* and the *higher-order indices*. The former describes the direct effect a factor has. The latter quantifies the interactions between the different factors that lead to an influence. For this, the variance V of the model is decomposed based on a *high dimensional model representation* with k being the number of input factors:

$$V = \sum_{i=1}^{k} V_i + \sum_i \sum_j V_{ij} \sum_i \sum_j \sum_k V_{ijk} + \ldots + V_{1,2,\ldots k}$$

It shows the variation decomposed into all dimensions of effects including the main effect and all interactions between the factors. Hence, the first-order sensitivity coefficient is defined as the extent of the main effect V_i in relation to the total variance V:

$$S_i = \frac{V_i}{V}$$

The higher-order effects are then defined similarly. Interesting is also the *total-order index* of a factor. It is the sum of the first-order effect and all higher-order effects in which the factor participates. Methods to compute such

measures of importance are, for example, the *(extended) Fourier amplitude sensitivity test (FAST)* [10] or the method of Sobol' [13].

2.3. Settings

The research in sensitivity analysis has shown that there are re-occurring questions and solutions that are summarised in so-called *settings*. There is a variety of those settings but two are the most common: (1) the factors prioritisation (FP) setting that ranks the factors in terms of their contribution to the variance and (2) the factors' fixing (FF) setting that is concerned with model simplification, i.e. which factors can be fixed without influencing the output. We are interested in both settings in the following. The FP setting can be answered using the first-order indices developed above. The first-order indices describe informally the expected amount of the total variance that would be removed if we knew the "true" value of that certain input factor. Hence, the factors with the highest first-order values are most important for further investigation.

The total-order indices as shown above describe the expected remaining variance if the input factor was varied but all other factors were set to a fixed value. This is the answer to the FF setting. Only input factors that do not significantly change the output when being varied can be safely removed from the model. Those factors can be fixed and nevertheless do not change the variance in the output.

3. The approach for software engineering

We propose in the following a way to use global sensitivity analysis for predictor models in software engineering. In particular, we describe three basic steps to analyse models by determining the distributions of the input factors, detecting errors using scatterplots, and quantifying the influence on the output by global sensitivity analysis. An overview of the three steps is also given in Fig. 2. As described above in Sec. 2, we assume that we have some model that computes a set of input factors X into one or more output factors Y.

3.1. Determining distributions

The first step in our approach is to determine the distributions of the input factors in X. This is needed to generate a large number of samples that are later used for the scatterplots and the global sensitivity analysis. For software engineering, there are mainly four ways to determine the distributions:

- Scientific literature
- Expert opinion
- Empirical project data
- Controlled experiments

What is the best way depends largely on the input factor itself. For some factors there are some specific empirical data that fits to the domain we want to analyse. Other factors might be well investigated in the scientific literature and a possible distribution can be found there. For some factors that are supposed to be very important it can even make sense to invest the effort for a controlled experiment. For empirical or experimental data, there are standard statistical methods to test for certain distributions and to estimate their parameters. However, there are often some factors that cannot or can only very costly be determined using those methods. Then assembling expert opinion is a reasonable way. This normally results in triangular distributions using the lowest, the highest and the most probable value for a factor. In summary, at the end of this step, we have a distribution for each input factor of the model.

3.2. Visualising using scatterplots

In the second step, sample data is generated and visualised by scatterplots. We introduced scatterplots as a method of sensitivity analysis in Sec. 2. They show the relationship of one input factor and the output factor. For this, we need two additional artefacts: (1) an implementation of the model and (2) a large number of sample data with the input and output of the model.

How the implementation looks like depends largely on the chosen tool support. The tool Simlab[1], for example, can work with any implementation as long as it can read and write a certain text format specified by Simlab. Hence, anything from C to Matlab is possible. This implementation is necessary to calculate the model outputs which is now important to be able to draw the scatterplot but also for the later analyses.

The sample data can be generated based on the distributions we determined above. There are standard methods (mainly Monte-Carlo methods) that sample the data following the specified distributions. The only input they need is usually the number of samples to be generated, a random seed and the planned analysis method. For scatterplots is the latter input not important. The Simlab tool is able to generate sample data for various analyses. This generated data needs to be input into the model implementation and the outputs need to be saved. Then scatterplots of each input factor and the output factor with all the samples should be made.

The use of the scatterplots is two-fold: (1) detection of errors and (2) first indications of influence. If there are er-

[1] `http://simlab.jrc.cec.eu.int/`

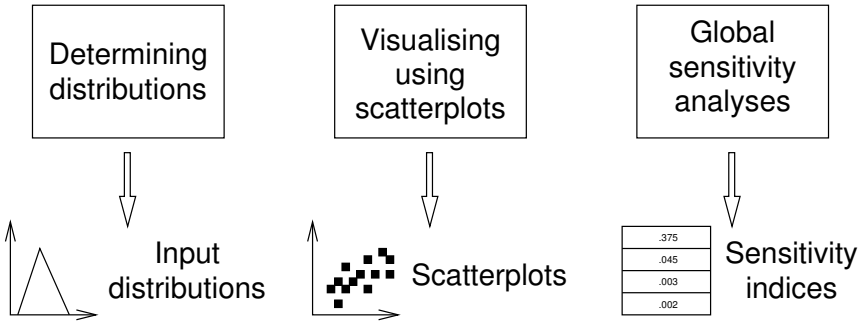

Figure 2. The three steps of the approach

rors in the model implementation or the distribution specification, they will be most likely to be seen in the scatterplots. Strange curves that suddenly change direction are good indicators for that. Typical scatterplots either look strongly chaotic or follow some kind of curve. A clear curve suggest a high correlation between the factor and the output and hence a possible high influence. Largely scatter points indicate that other factors have stronger influence. To summarise, the result of the scatterplots are assurance that there are no errors and first indications of factors with a high influence.

3.3. Applying global sensitivity analysis

For the global sensitivity analysis, there are again several possibilities but we suggest to use the FAST method as described in Sec. 2. The result of the FAST method are first-order and total-order indices that describe the quantitative difference between the input factors. This allows an easy evaluation of the questions typically asked such as which factors can be removed from the model.

For the execution of the FAST method, we have everything at hand from the first two steps. We specified the distributions of the input factors, built an implementation of the model to be analysed and generated sample data. It is only important that by generating the sample data, we set the FAST method as aim. Then we can use a sensitivity analysis tool such as Simlab to calculate the sensitivity indices. These indices need to be interpreted in the following.

The first-order indices give the share of the output variation that is directly related to each input factor. For example, a first-order index of 0.2 means that the input factor contributes directly to 20% of the variance of the output factor. The sum of these first-order indices is always ≤ 1. The difference to 1 is the amount of higher-order effects, i.e. interactions between the input factors. Hence, the interpretation is that by reducing the variance in an input factor by determining it more precisely, we can reduce the amount given by the first-order index in the variation of the output. This is exactly the solution for the factors priorisation setting.

The total-order indices describe the share of the output variation that is *related* to each input factor. This includes all interactions. A total-order index of 0.12 means that 12% of the output variation is somehow caused by this input factor. This includes the direct effect as well as interactions with other factors. Therefore, the sum of these indices is usually greater than 1. The interpretation is that by removing this factor we remove the amount of the total-order index from the output variation. Hence, we can only remove factors with very small total-order indices so that the output is not changed significantly. A total-order index below 0.1% is typically considered very small. This is the solution to the factors fixing setting.

4. Example

The approach described above is now applied in an example. The predictor model we analyse is a software reliability model that is used in practice.

4.1. Model under analysis

The model under analysis is a software reliability growth model that is used to predict the failure rate and the cumulated number of failures of a software system. It was developed in cooperation with the communication networks department of the Siemens AG. The general idea in the model is that the relationship between the failure rates of the individual faults follows a geometric progression. This geometric progression was derived by analysing old project data at Siemens.

In the paper by Wagner and Fischer [17], this was developed into a stochastic reliability model that has essentially four input factors:

1. d: The exponent of the geometric progression
2. p_1: The highest failure rate of a fault

3. t: The execution time for which we want to predict

4. *inf*: An approximation of infinity

The last factor *inf* is necessary because the model assumes that there are potentially infinitely many faults in the software. However, this prevents an actual calculation and therefore this approximate value is used. The output value used in the following is the number of failures up to time t denoted by μ.

There are three questions w.r.t. this model that are interesting for its use in practice:

1. In how much detail does the execution time needs to be measured?

2. Can the factor *inf* be set to one value without influencing the output significantly?

3. Which of d and p_1 is more important to estimate early based on other factors?

The first question is based on the observation that for software reliability only execution time is sensible. However, it is often difficult to estimate how frequently the software is used in the field by different customers. Therefore, the estimates can vary and the question is whether this can be a problem. The second question aims at simplifying the model. By now it is necessary to estimate the highest possible number of faults in order to use the model. It would be helpful to know that *inf* can be set to one value without significant impact. Finally, the third question is rooted in the desire to predict reliability early in the development. Hence, we investigate whether other factors that are available early can be used to estimate the input factors of the model. Hence, we need to know which of d and p_1 is more beneficial to estimate.

4.2. Scatterplots

A preliminary step in sensitivity analysis is to visualise the relationship of the input parameters and the output using scatter plots. We use the model as described in Sec. 4.1 and a set of data generated by the Monte-Carlo method. For the sake of simplicity we already make use of the same set that is used in Sec. 4.3 for further analyses. It analyses the case where we want to know the number of cumulated failures at about $t = 100,000$. The sample size is 1,988.

Execution time. The first model parameter we analyse is the execution time t. It is one of the parameters that is not estimated by a Least Squares method but needs to be determined based on the usage of the software. Hence, there is also uncertainty because we might not document the usage accurately or we might not know when exactly the software was put into operation by a customer. Hence, we model the input parameter t using a normal distribution with $\mu = 100,000$ and $\sigma = 200$. The scatter plot is shown in Fig. 3.

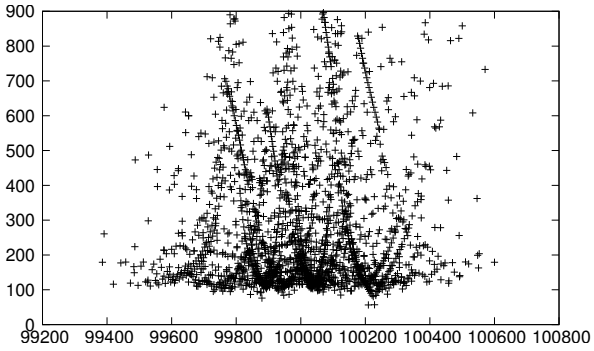

Figure 3. A scatterplot of the relation of t and μ

We observe that there is no straightforward relationship between the execution time and the number of failures. We can sense the normal distribution in the data but in general it is more a cloud. This suggests that the variation in t is not important for the model output. Moreover, we observe that independent of the variation in t the most points lie between 100 and 200 which seem to be the most probable outcome and that below 100 there are only few data points.

Complexity. The parameter d of the model is supposed to represent the complexity of the software. As we do not have much further knowledge on how this representation can be traced back to direct metrics of the software, we assume a uniform distribution. Earlier experience with the model however has shown that only values in $[0.9, 1.0[$ make sense. Hence, we restrict the distribution to this range. The scatter plot is depicted in Fig. 4.

The observations are quite different to the relationship between t and μ. There seems to be a strong direct relationship between d and μ, at least until about 0.98. We emphasised this with the vertical dashed line. After that point the relationship is less clear. Probably, some higher order effects take place in that area. However, over the whole range, d describes an upper limit of mu. Also after 0.98 there seems to be a lower bound a bit lower than 400 (horizontal dashed line).

Highest failure probability. The other main parameter of the model is the highest failure probability of a fault p_1. We have no earlier knowledge about its distribution. Hence, we distribute it uniformly over the whole range $]0, 1[$ of a probability value. We depict the scatter plot in Fig. 5.

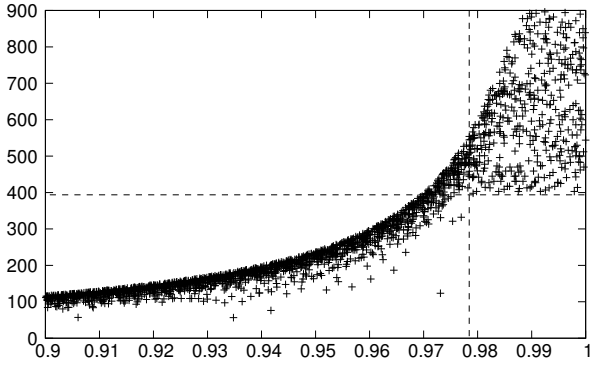

Figure 4. A scatterplot of the relation of d and μ

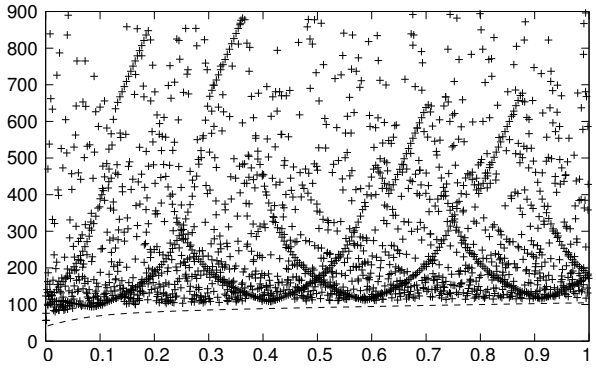

Figure 5. A scatterplot of the relation of p_1 and μ

Similar to t, there is no clear relationship visible between p_1 and μ. There are some slight indications of curves but generally the points seem to be randomly distributed. The curves are probably caused by the normal distribution of t. We see again the concentration between 100 and 200 independent of the value of p_1. This suggests again a rather small impact on the output. As opposed to d where we saw an upper bound, p_1 rather shows a lower bound at about 100. We indicated the lower bound by a dashed line. This lower bound seems to be an exponential relationship because we have lower values only left of 0.1.

Approximation of infinity. Finally, we also looked at the approximation of infinity that must be introduced as an additional parameter when realising the model in software. This is because we add the probabilities of all the faults where there is the assumption that there are infinitely many faults. As there is no closed expression to do that in our case, we need to introduce a high value that acts as an approximation. Hence, it is interesting to know how much influence this parameter has or if it is possible to just fix it at some value. We do not have much experience with this parameter but for the chosen execution time we think that about 500 faults are reasonable. To further analyse the parameter we modelled it as uniformly distributed between 400 and 900. We show the relationship for this in Fig. 6.

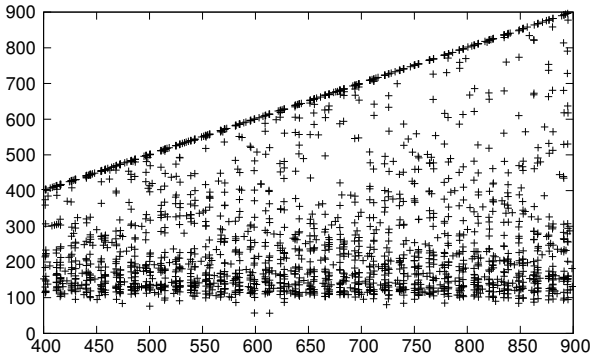

Figure 6. A scatterplot of the relation of *inf* and μ

The most obvious relationship is the linear upper bound that is given by *inf* itself. This is not surprising as our model is a kind of counting process. Each fault can result in a failure at most once and hence there can be no more failures than there are faults. The highest possible number of faults is given by *inf*. Apart from that we also see the most data points between 100 and 200 and a weak lower bound at about 100.

4.3. FAST

Moving a step forward from scatter plots, we now quantify the influence of the input parameters on the output. For this, we look at both settings: (1) factors priorisation and (2) model simplification. We use the FAST method for calculating the first- and the total-order indices in order to give answers w.r.t. the settings. The results are shown in Tab. 1.

We observe that in the first-order indices, d has by far the greatest influence on the output with nearly 91%. The factors p_1 and *inf* are far less influential with around 1%. The execution time t is not significant. To follow up on the questions posed in Sec. 4.1, we can answer the third question based on these results. The factor d is the most beneficial to estimate. Note, that this corresponds to the first observations in the scatterplots above. The clearest relationship was between d and the output μ that also has the highest index. Apart from the first-order effects there remain higher-order effects of 0.06788. Their influence is visible in the total-order indices.

The ranking of the total-order indices is similar to the first-order indices. However, we look at the low values here

Table 1. Sensitivity indices

First-order indices	
d	0.9096
p_1	0.0142
inf	0.0077
t	0.00062
Total-order indices	
t	0.006194
p_1	0.034642
inf	0.074103
d	0.976614

because we need to know which factors have only a small influence. In this case, the variation of t has no significant influence on the output with below 1%. The factors p_1 and d lie again in the middle with about 3% and 7%, respectively. Also in total, d has the most influence with over 97%. For the first question from Sec. 4.1, we can answer that the variation in t is not significant. Hence, more detailed estimates are not necessary. The answer to the second question is that the factor *inf* is significant enough to not be set to one single value. It needs to be estimated for each case.

5. Further applications

We used a preliminary version of the approach described in this paper for the cost-benefit model proposed in [15]. The model describes the economics of defect-detection techniques based on a large set of parameters that describe the defects contained in the software as well as the used defect-detection techniques. This huge amount of parameters (41 for a small project) in the original model made its application costly. Hence, it is a perfect candidate for sensitivity analysis. We determined the distributions of the input factors based on an empirical meta-analysis [14]. The results of the analysis included that two of the parameters are most beneficial to determine in more detail. We also identified some parameters that were fixed in a more practical model. However, the second-order indices indicated that these were influential so that we lost precision.

We applied the global sensitivity analysis method also to the effort prediction model COCOMO [2]. This work has not been published so far. We used a publicly available data set from NASA projects in order to determine the most important factors in their project contexts. We found again that from the 18 input parameters (development modes and cost drivers) only one factor has an extremely strong first-order effect: the estimation of the lines-of-code of the software to be built. Hence, it is most beneficial to have a good estimate of this parameter. The picture is not as clear for fixing parameters. Most of the parameters have a total-order effect of 11–16% and hence are significant enough to not be removed from the model.

Our colleagues Deissenboeck and Pizka used global sensitivity analysis on a process simulation model in [4] in order to analyse which transition probability in the Markov chain used for modelling the process is most important. This parameter was then manipulated in order to evaluate whether the general conclusions from the simulation (that one process variant needs less effort) changes.

6. Related work

Rodrigues, Rosenblum and Uchitel [8] made sensitivity analyses of a scenario-based reliability prediction model. They had the aim to investigate the effect of component reliability and usage profiles. For the analysis, they only varied the respective parameter and set the others to 1. This does not allow to analyse higher-order effects. In global sensitivity analysis all parameters are varied simultaneously to circumvent that effect.

Sensitivity analysis has already been used for other software engineering models. For example, Boehm made a sensitivity analysis of the COCOMO model in [2]. However, those analyses have a slightly different aim and do not use global methods that include the whole distribution in the analysis. Freimut, Briand, and Vollei [5] used sensitivity analysis on a cost model for inspections using data from expert opinion. This shows that sensitivity analysis is a valid tool for such models. However, the used sensitivity analysis methods are correlation-based and do not offer the benefits of the global methods shown in this paper. Most importantly, global methods do not have the risk of overemphasising local effects.

In general, Kitchenham et al. [6] suggest using sensitivity analysis as part of empirical research. However, they do not elaborate on the methods to be used and many concentrate on identifying outliers.

7. Conclusions

The use of predictor models is often difficult and elaborate due to the number of input factors involved. We propose an approach based on the established technique of *global sensitivity analysis* in order to analyse the importance of the input factors. The approach is *model-free*, i.e., it does not depend on certain properties of the model. Furthermore, it is *global* in the sense that it incorporates the whole distributions in the analysis. The only two prerequisites to use the approach is (1) information about the distribution of the input factors and (2) an implementation of the model. The distributions can be determined using various methods including empirical data or expert judgement.

We have made promising experience using this approach so far. The example above shows that it is also useful for the analysis of reliability growth models in order to determine the importance of the factors and hence to optimise the measurement effort. Furthermore, we were able to analyse and simplify a cost-benefit model of quality assurance [15] and other models so that we are confident that the approach is generally applicable for predictor models. Nevertheless, we plan to investigate further models for which we are able to collect the necessary data.

Acknowledgements

We are grateful to the organisers and lecturers of the *Summer School on Sensitivity Analysis of Model Output* 2006 in Venice, Italy that showed us the many possibilities of sensitivity analysis. Furthermore, we would like to thank Sebastian Winter for useful comments.

References

[1] R. D. Banker, S. M. Datar, C. F. Kemerer, and D. Zweig. Software Complexity and Maintenance Costs. *Communications of the ACM*, 36(11):81–94, 1993.

[2] B. W. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.

[3] K. Chan, S. Tarantola, A. Saltelli, and I. M. Sobol. Variance-Based Methods. In Saltelli et al. [11], pages 168–197.

[4] F. Deissenboeck and M. Pizka. The Economic Impact of Software Process Variations. In *Proc. International Conference on Software Processes (ICSP '07)*. Springer-Verlag, 2007.

[5] B. Freimut, L. C. Briand, and F. Vollei. Determining Inspection Cost-Effectiveness by Combining Project Data and Expert Opinion. *IEEE Transactions on Software Engineering*, 31(12):1074–1092, 2005.

[6] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Eman, and J. Rosenberg. Preliminary Guidelines for Empirical Research in Software Engineering. *IEEE Transactions on Software Engineering*, 28(8), 2002.

[7] J. D. Musa. *Software Reliability Engineering: More Reliable Software Faster and Cheaper*. AuthorHouse, 2nd edition, 2004.

[8] G. N. Rodrigues, D. S. Rosenblum, and S. Uchitel. Sensitivity Analysis for a Scenario-Based Reliability Prediction Model. In *Proc. Workshop on Architecting Dependable Systems (WADS '05)*. ACM Press, 2005.

[9] A. Saltelli. What is Sensitivity Analysis? In Saltelli et al. [11], pages 4–13.

[10] A. Saltelli and R. Bolado. An Alternative Way to Compute Fourier Amplitude Sensitivity Test (FAST). *Computational Statistics & Data Analysis*, 26(4):445–460, 1998.

[11] A. Saltelli, K. Chan, and E. M. Scott, editors. *Sensitivity Analysis*. John Wiley & Sons, 2000.

[12] M. Shepperd and C. Schofield. Estimating Software Project Effort Using Analogies. *IEEE Transactions on Software Engineering*, 23(11):736–743, 1997.

[13] I. M. Sobol. Sensitivity Analysis for Non-Linear Mathematical Models. *Mathematical Modelling and Computational Experiment*, 1:407–414, 1993.

[14] S. Wagner. A Literature Survey of the Quality Economics of Defect-Detection Techniques. In *Proc. 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE '06)*, pages 194–203. ACM Press, 2006.

[15] S. Wagner. A Model and Sensitivity Analysis of the Quality Economics of Defect-Detection Techniques. In *Proc. International Symposium on Software Testing and Analysis (ISSTA '06)*, pages 73–83. ACM Press, 2006.

[16] S. Wagner. *Cost-Optimisation of Analytical Software Quality Assurance*. PhD Dissertation, Technische Universität München, 2007. To appear.

[17] S. Wagner and H. Fischer. A Software Reliability Model Based on a Geometric Sequence of Failure Rates. In *Proc. 11th International Conference on Reliable Software Technologies (Ada-Europe '06)*, volume 4006 of *LNCS*, pages 143–154. Springer, 2006.