Simulation of Experiments for Data Collection – a replicated study

Per Runeson and Mattias Wiberg
Dept. Communication Systems
Lund University, Box 118, SE-221 00 Lund, Sweden
per.runeson@telecom.lth.se

ABSTRACT

Simulations can be used as a means for extension of data collection from empirical studies. A simulation model is developed, based on the data from experiments, and new data is generated from the simulation model. This paper replicates an initial investigation by Münch and Armbrust, with the purpose of evaluating the generality of their approach. We replicate their study using data from two inspection experiments. We conclude that the replicated study corroborates the original one. The deviation between the detection rate of the underlying experiment and the simulation models was 2% for the original study and is 4% in the replicated study. Both figures are acceptable for using the approach further. Still the model is based on some adjustment variables that are not directly possible to interpret in terms of the original experiment, and hence the model is subject to improvement.

1. INTRODUCTION

To conduct empirical studies in software engineering costs resources, as they always include human subjects. This is unavoidable, as we want to study how the subjects behave. However, we may possibly utilize the information better, that is gained in empirical studies. Simulations have appeared as a vehicle to achieve this [13]. Münch and Armbrust [11][2] have conducted a study to evaluate the principle of collecting empirical data from simulations. Software inspections are rather well investigated empirically [3], but many other areas in software engineering lack empirical data. The access to empirical data makes the area of software inspections suitable for evaluating the simulation principles, and then the experiences may be transferred to other software engineering domains with less empirical data available.

Basically, the approach is to build a simulation model based on data from experiments, and evaluate how well the simulation model resembles the empirical studies. If the simulation model resembles the empirical study, we may alternate the model settings and investigate alternative variants in what-if scenarios. Further, the simulation model may be used to increase the statistical significance of empirical data, using a bootstrap approach [9]. The results of the Münch and Armbrust study are encouraging, as the deviations between the empirical studies and the simulated results are quite small.

The purpose of this study is to investigate whether the original results are due to specific characteristics of the data, or they are more general. In order to investigate this, we have conducted a replication of their study. The same simulation model is implemented in a simulation tool, calibrated using data from two experiments on inspections [14][15], and then evaluated. Another simulation tool was used, primarily due to limited access to the original toolset. The mean deviation between the real data and the simulation data is less than 4% in the replicated study. This is more than the reported 2% deviation in the original study, but still an acceptable result. Hence we conclude that the simulation model seems to be general enough to capture the behavior of other inspection experiments. We also extend the model to capture some random variation in reviewer performance.

The paper is outlined as follows. In Section 2, we provide some background on process simulation. The goal of the study and the methodology used are presented in Section 3. The simulation model with its parameters is presented in Section 4, while the results of the replication are reported in Section 5. Finally, we summarize our conclusions in Section 6.

2. SOFTWARE PROCESS SIMULATIONS

Software process simulation is a way of supporting decision making in a software development process. Simulations can be used for a wide variety of applications in a software development process, ranging from smaller parts like a single inspection to larger ones concerning a whole organization.

The basis for simulations is a computerized abstraction of the real life process, a simulation model. When developing the simulation model the goal is to address the significant features of the real life process. Kellner et al. stress three different aspects to consider in building simulation models, namely 1) the purpose, 2) the structure of the model, and 3) the implementation [10].

2.1. Purpose

Several issues in a software development process can be dealt with using simulations. Kellner et al. [10] divide them into six categories: 1) Strategic management, 2) Planning, 3) Control and operational management, 4) Process improvement and technology adoption, 5) Understanding and 6)Training and learning.

The strategic management category involves questions like "*How would this process improvement affect our productivity over the next two years?"* while in the process improvement category the issue is the process improvement itself, like *"how efficient is the new testing technique?"* In the planning category the aim is to solve problems concerning cost/benefit predictions, whereas a control and operational management question is, "*When should we stop testing?*" Finally, apart from supporting decision making, simulation can help in understanding complex process flows, aid communication and act as a learning tool of the software process [1][5].

2.2. Structure

There are four areas to consider when defining the simulation model structure [10]: the scope of the model, the result variables, the abstraction level of the model, and the input parameters.

Model scope. Scoping of the model involves setting the limitations for what the model should comprise. A simulation model cannot include everything that might possibly impact on the result variable, but must be limited to the major factors. To be able to select a scope it is important to understand the purpose of the model i.e. which questions need to be answered.

Key result variables. When deciding which result variables a model should provide, the purpose of the model and the questions attached to it, are once again decisive. Result variables may be cost/effort, defect level, staff utilization rate, cost/benefit and throughput/productivity.

Process abstraction. The issue with process abstraction is to consider the tradeoff between complexity and data sufficiency, and then to choose which key elements of the process that will make the model useful. Important elements to identify could be key activities, primary objects, vital resources and decision points [10]. The process abstraction aspect is highly dependent on the selection of both the result variables and the input parameters.

Input parameters. As Raffo and Kellner [12] point out, the results from a simulation model can only be as accurate as the input supports. In addition to this it is naturally also important to select the input parameters that are essential for the model purpose and key result variables.

All these factors, and also the model purpose, will affect the model behavior and thus also each other. If one factor is modified, the rest will most likely have to be reviewed once again. In conclusion, model building is an iterative process with the aim of finding an adequate representation of a real life process.

2.3. Implementation

When the purpose, scope, key parameters and abstraction level of the simulation model are established, the final question is how to implement the simulation model. This involves the decision of simulation approach and consideration of available techniques and related simulation tool.

Simulation approach. Kellner et al. [10] list a number of the simulation approaches applied to software processes: 1) State-based process models, 2) Discrete event simulations, 3) Continuous simulations, 4) Hybrid simulations, 5) Rule-based languages, and 6) Queuing models.

They emphasize that no modeling approach or tool is the best one for every situation. However, in their study they provide general guidelines for the selection. Continuous simulations are suggested to be suitable for high-level analysis, like strategic management and for processes with longer time-span. Discrete event and state-based simulation is on the other hand well-suited for a detailed process analysis.

Additionally, in discrete event models the simulation moves forward in discrete steps which allow the process and attributes to be analyzed in detail. This is not the case in a continuous model where the dynamics take place in continuous time and process details can not be individually traced.

Simulation techniques. The simulation model can be developed using visual or textual techniques. Visual models have become the norm in software process simulations [10], because they are easier to understand.

Furthermore, the input parameters used in a simulation model can be deterministic, stochastic or a combination of both. Deterministic parameters have single values without variation, whereas the stochastic ones are random numbers from a probability distribution.

The decisions concerning simulation approach and technique highly affect the choice of simulation tool. The tool must be able to support the chosen approach and its properties. Furthermore, it must also provide the adequate techniques for building the model. This means, that in addition to the approach, the tool must also support the implementation of key parameters and abstraction level.

3. STUDY DEFINITION AND METHODOLOGY

The *purpose* if the simulation study reported in this paper is to understand the characteristics of a software inspection process, with the intention of using the simulation model for empirical data collection. In particular, replication methodology is used to investigate the generality of the simulation approach.

The *scope* of the simulation model is inspection processes, as defined by the original study [2][11]. The primary result variable is the number of defects found in the inspection. We abstract the model to capturing artifacts and reviewers, with attributes qualifying each instance. The input parameters are artifact and reviewer attributes, taken from experimental studies on software inspection.

The *implementation* of the simulation model is based on the continuous simulation approach. The model is implemented in the MATLAB/Simulink environment, which has a graphical representation of model building blocks.

The overall goal of the study is to investigate whether the simulation model is specific for the inspection experiments on which it was built, or it is more general to its character. A secondary goal is to evaluate the feasibility of using the MATLAB/Simulink environment for process type simulations.

The replication study is conducted in four main steps:

1. Implementation of the model in a simulation tool
2. Validation of the model implementation, using the original data
3. Replication of the study, using the new data
4. Extend the model for better performance, and evaluate

The implementation of the model was quite straightforward, as we got access to all relevant information from the original study [2][11]. We chose to implement the model in the MATLAB/Simulink[1] environment instead of the original Extend[2] environment, primarily due to better access to the tool. The implemented model was validated using one data set from the original study to ensure that the new implementation of the model is identical to the original implementation. The new implementation provided exactly the same output with the original data, as the original study, and hence we consider the two implementations being identical.

Finally, the behavior of the model was validated using empirical data from two experimental studies conducted at Lund University [14][15]. Both studies concern inspection of design documents, comparing two different reading techniques. One of the treatments in both studies is usage-based reading, guided by prioritized use cases. The other treatment is usage-based reading with randomized ordered use cases [14] and checklist-based reading [15] respectively.

The extension of the simulation model is conducted as a separate step, and the evaluation of the extensions is conducted, using the same data as the evaluation of the replication.

4. MODELS

4.1. The original study

The purpose of the original model was to investigate how well a simulation model captures the behavior of a software inspection. The structure of the simulation model developed by Münch and Armbrust [2][11] is a discrete event model of the factors influencing on the inspection performance, with the dynamic behavior modeled as influence diagrams. They implemented the model in a modeling and simulation environment, called Extend.

The dynamic simulation model is based on a static process model of the actual inspection process, presented in Figure 1. The development process as such is not implemented in the simulation model, but it is designed based on the artifacts and activities of the process, their attributes and the relations between these attributes. The relations are presented in an influence diagram in Figure 2, which is used to model the dynamic behavior of the inspection process.

The static inspection model contains an overview of all the steps, roles and artifacts in the process. The inspection steps, *Planning, Overview, Preparation, Inspection Meeting, Rework* and *Follow-up* are the basis for the model. In addition to the steps, the roles in the process are also displayed, which are *Moderator, Designer, Implementer, Reader* and *Tester*. Each role has some specific skill or knowledge and the team members can play multiple roles. In the static model, the roles and not the actual team members, are displayed. Münch and Armbrust use a somewhat different terminology regarding role names, but the function of the roles are the same. The *Inspector* and the *Author* are for example simply roles played by several team members. The artifacts that move through the inspection process are also displayed in the inspection process in Figure 1 [11]. These include the actual design document and the documents produced during the inspection, like a defect list.

[1] http://www.mathworks.com
[2] http://www.imaginethatinc.com

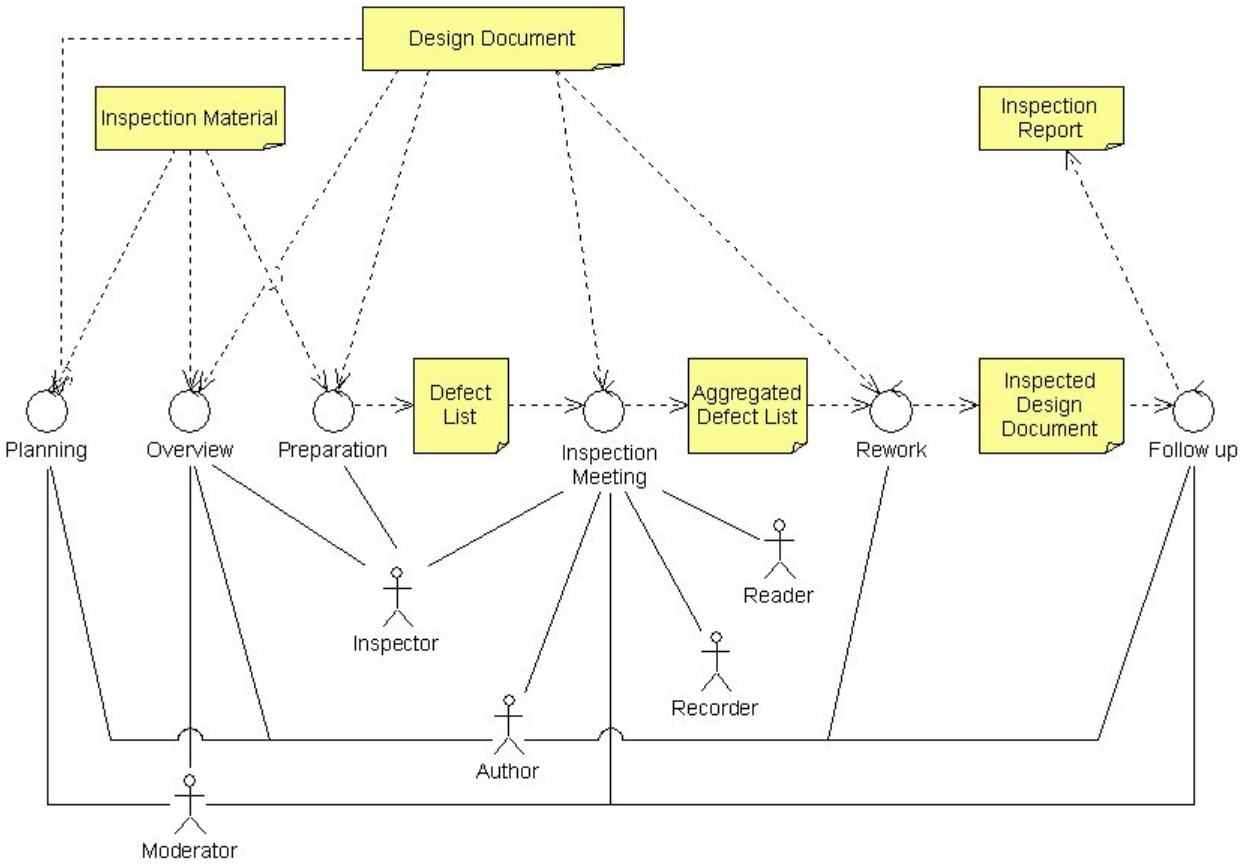

FIGURE 1. Static inspection process model [11]

To provide a better understanding of the inspection process model, the steps are described below along with the roles and artifacts involved. The *planning* step involves the forming of the inspection team and the assignment of the roles. This is organized by the moderator who functions as a coach for the entire inspection. The *overview* step is an optional one, in which the author may present the artifacts and additional properties of the software product. After this the team members individually inspect the artifacts in the *preparation* step. This generates lists from the reviewers which are compiled into a *defect list* by the moderator. With the defect list as a basis, the *inspection meeting* is conducted with all the team members present. In the inspection meeting the defects are identified and documented. Solutions or suggestions of implementation are not to be discussed. An *aggregated defect list* is compiled by the moderator and provided to the author. The author performs the *rework*, i.e. corrects the defects in the list and delivers an *inspected design document*. If an artifact contained many defects the decision of a reinspection and further rework can now be taken. Finally, in the *follow-up* step the moderator checks that the issues in the defects list are resolved.

The purpose of the simulation model is to be a decision support for planning in software projects. Furthermore, the model development was focused on the integration of empirical data. This means that some aspects of the inspection process can be excluded from the simulation model and thereby reducing its complexity. The simulation model is still useful in the intended context and the data reduction limits the scope and consequently also the ab-

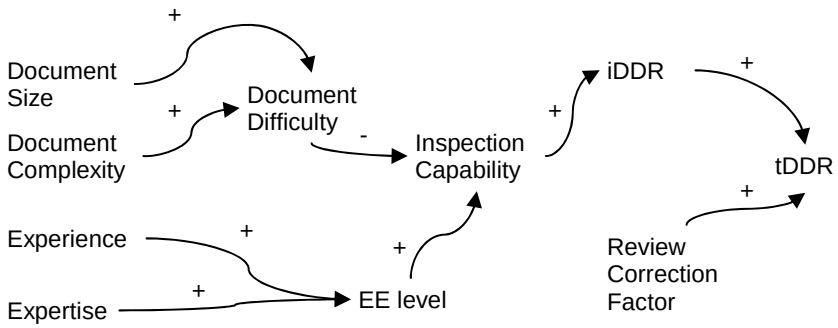

FIGURE 2. Influence diagram [2]

straction level. In conclusion, the simulation model abstracts from several details in the inspection process, but the key elements that will make the model useful are all included.

The key factors, chosen by Münch and Armbrust [11] to be the most influential to the inspection process performance, are the capacity of the available reviewers and the status of the document to be inspected. The two factors, *document* and *reviewers*, are characterized by a number of attributes. The factors, the attributes and a short description of them are presented in Table 1.

The influence diagram in Figure 2 describes the interaction between the attributes in Table 1. The diagram and the static model were used as a basis for the dynamic model, which in the original study was implemented in the simulation environment Extend as a discrete event model. For calibration and further development, Münch and Armbrust integrated empirical data from two experimental studies [4][8]. Finally, a validation of the simulation model was performed using a data set from Biffl and Gutjahr [6].

The output of the simulation model is the average team defect detection rate (*tDDR*), i.e. the rate of the defects found by the simulated teams. The simulation model is evaluated with respect to its deviation from the value achieved by constructing virtual teams (VT) from the original experiment. A virtual team of size *N* is composed by drawing *N* individual reviewers from the data set, and calculating the *tDDR* for that team. All combinations of reviewers are composed into teams on *N* reviewers, and statistics on the *tDDR* can be calculated. See [7] for further details on VT or Virtual Inspections. Virtual teams is a variant of bootstrapping [9].

TABLE 1. Model attributes

Factor	Attribute	Description
Document	Number of defects	Total number of defects in the document
	Size	Document size
	Complexity	Document complexity
Reviewers	Team size	Number of reviewers
	Experience	Reviewer experience with reading technique, document type etc.
	Expertise	Reviewer domain expertise
	iDDR	Individual defect detection rate
	Document coverage	Percent of document covered by reviewer

The deviation is defined as the average defect detection rate of the virtually composed inspection teams, minus the simulated team detection rate ($tDDR_{\Delta} = tDDR_{VT} - tDDR_{sim}$). The results of the Münch and Armbrust study show quite a small deviation between the virtual teams of the empirical data and the results from the simulation. The mean deviation in their study is about 2%.

The principal procedures of the original study are summarized in Figure 3 (left). The model is built and calibrated using information from the experiments. The individual detection rate (iDDR) is fed into the model, and the team detection rate (tDDR) is calculated from the simulation. The outcome is evaluated by calculating the deviation from the experimental value.

4.2. The replication study

The purpose of the replication is the same as in the original study, i.e. to investigate how well a simulation model can capture the behavior of a software inspection experiment. Now we add the aspect of generality in that we apply the previously derived model to new data. The structure of the model is the same as in the original study, while the implementation is conducted in another toolset, for practical reasons. The main difference is that we use data from another set of experiments as calibration and input to the model, as depicted in Figure 3 (right). We also investigate some extensions to the simulation model.

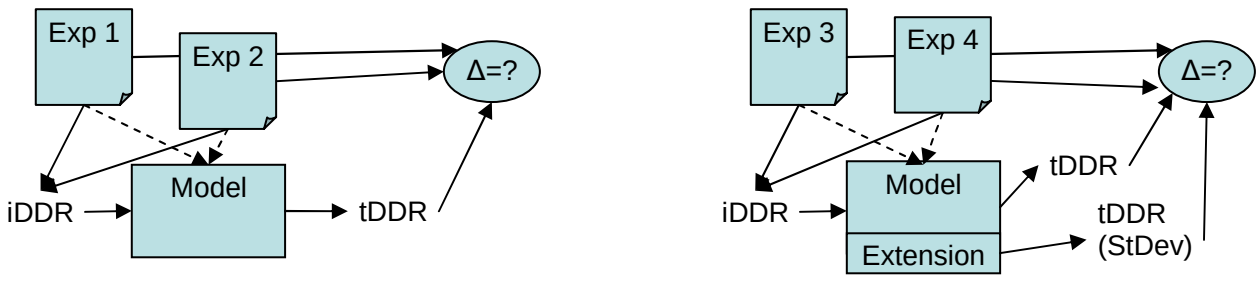

FIGURE 3. Overview of original study procedures (left) and replication study procedures (right)

4.2.1. Model implementation

The MATLAB/Simulink simulation environment differs slightly from Extend, used in the original study, which required a different approach to some modeling concepts. However, the differences were mainly implementation details, e.g. other building blocks and changes in representation of the model properties and attributes. With access to relevant information about Extend and the original model, these implementation difficulties could be resolved. The basic structure and principles in the replication model and the original model, are identical.

The input data are read from file to provide an efficient setup for simulation execution. The file contains attributes which are here explicitly set for each reviewer.

The first step in building the simulation model was to implement the basic structure, i.e. attribute realization and their connections. The factors and attributes described in Table 1, their interaction extracted from the influence diagram and other additional calculations were the basis for this. In addition to the listed attributes, the original study provides tree adjustment variables. These variables give further control of the model behavior and consequently a better description of the inspection context. The adjustment variables are *Experience Importance, Expertise Importance* and *the Review Correction Factor*. The adjustment variables and the attributes represent the input variables to the simulation model and are briefly described below.

4.2.2. Document attributes

The documents used in the simulation model are design documents characterized by the attributes described below. These can be found as boxes in the top-left corner of Figure 4.

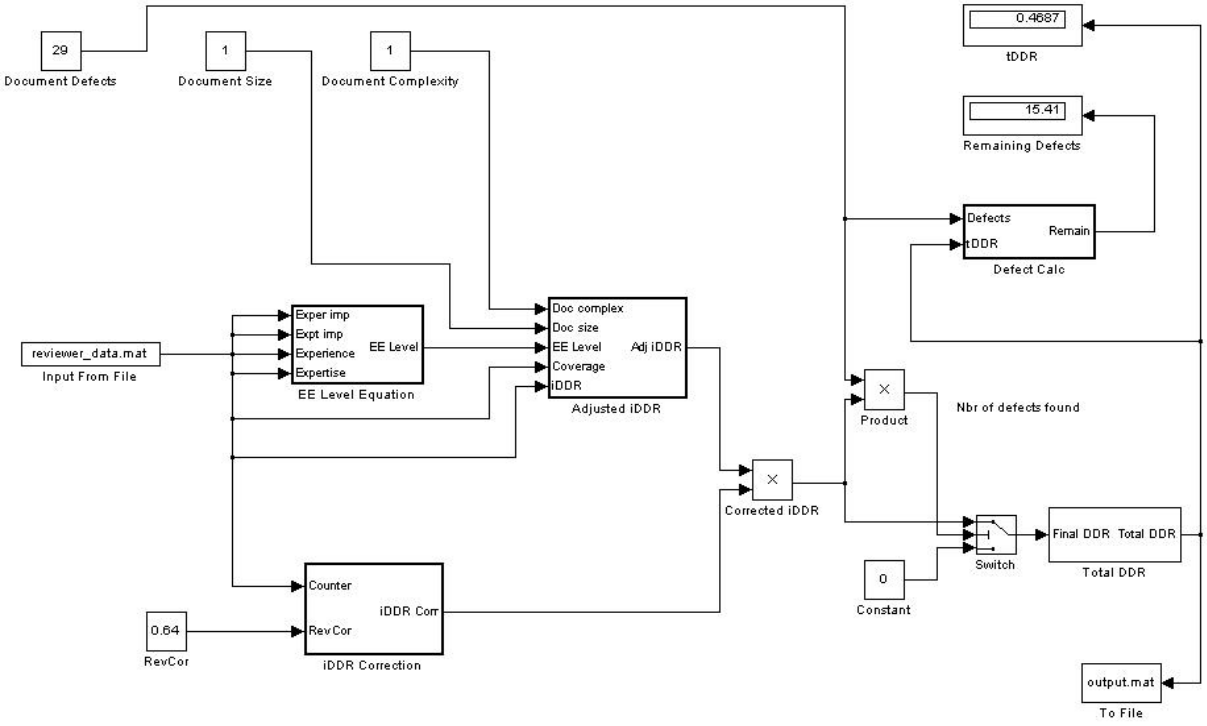

FIGURE 4. The simulation model expressed in MATLAB/Simulink.

- *Document Defects* is the total number of defects that exist in the document. The attribute is used in the model to display the effects of the inspection process, i.e. the reduction of defects. The attribute is simply implemented as a constant and can be set to values in the interval [1,n].
- *Document Size* is the attribute that reflects the size of the document. The attribute is implemented as a constant and can be in the interval [0,2], where the value 1 represents an average document size in the context where it is used. Values over or below this average reflects a larger or smaller document. In the original study, the average value is a document with 17 pages.
- *Document Complexity* is designed to reflect the complexity of the document and is also implemented as a constant with values in the interval [0,2], with 1 as the average. Like Document Size, the attribute can assume values over or below the average to display complexity different from the usual.

4.2.3. Reviewer attributes

The individual reviewers form an inspection team and inspect the design documents. The reviewers are described by the attributes below. These are input to the simulation model in a file called *reviewer_data.mat* in Figure 4.

- *Team size* is the number of reviewers contributing in the inspection.
- *Experience* reflects the experience a reviewer has with the inspection process, the document type and the reading technique. The attribute can be set to values in the interval [0,2], with 1 as the average.
- *Expertise* describes what domain knowledge the reviewer has. Values are in the interval [0,2].
- *Document Coverage* reflects how the reviewer makes use of the assigned inspection time. Values are in the interval [0,1] and 1 indicates that the entire document was reviewed.
- *Individual Defect Detection Rate (iDDR)* describes the reviewer performance, i.e. how large share of the defects a reviewer found of the total amount contained in a document. Values are hence in the interval [0,1].

4.2.4. Adjustment variables

In addition to the attributes, the original model has a set of adjustment variables, which take some specific inspection considerations into account.

- *Experience Importance* gives the possibility to adjust the model to an inspection process where the experience of the reviewers is of special importance.
- *Expertise Importance* is used to reflect if an inspection process is highly dependent on reviewers with special domain experience.
- *Review Correction Factor (RevCor)*. When an inspection is performed by a team of reviewers some defects will be found by more than one reviewer. This will create an overlap which can differ with different inspection techniques. The overlap in the model is reflected by the Review Correction Factor. Values are in the interval [0,1].

4.2.5. Model validation

In order to confirm that the new implementation of the simulation was correct, the replication model was validated with the data from the original study. The replication model was run with data sets used for calibration by Münch and Armbrust. The data originates from two experimental studies [4][8].

The validation was performed by setting up the replication model with the data and inputs from the original simulation, execute a simulation run and finally comparing the outcome *tDDR* with the *tDDR* from the original study. The simulation setup is presented in Table 2.

TABLE 2. Original simulation setup, used in replication model validation

Attribute	Value
Number of defects	28
Document size	1
Document complexity	1
Team size	3
Reviewer experience	1
Reviewer expertise	1
iDDR	0.2464; 0.3214; 0.2133; 0.2593; 0.1459; 0.2246
Document coverage	1

The simulation model was run with six different values of the average *iDDR* from the original study data sets. The output from the simulation, the original *tDDR* and the replication model *tDDR*, were identical and the replication model was therefore considered to be identical to the original model.

4.2.6. Replication

The data used for the replication originates from two experimental studies. The first study [14] evaluates the use of usage-based reading (UBR). The experiment in the study was conducted on 27 third-year software engineering students and was designed to compare prioritized to randomly ordered use cases in UBR. The experiment had two groups, one with 13 students using randomly ordered use cases and one with 14 students using use cases in prioritized order. A questionnaire was used to investigate the expertise and experience among the students and from the result it was concluded that the subjects, although being students, could be considered to be similar to software developers. The document that was inspected is a design document with 9 pages which contains 37 defects. A defect is present when some part of the design differs from the stated requirements. These data sets are denoted *Rand* and *UBR1* respectively.

The second study, also presented by Thelin et al. [15], describes an experiment that compares usage-based and checklist-based reading, i.e. UBR and CBR. The participants in the experiment in this study were 23 fourth year software engineering students and could, like in the experiment above, be considered similar to software developers. The experiment had two groups, one with 12 students using CBR and on with 11 students using UBR. The inspected document is a design document with 9 pages which contains 38 defects. A defect is present when some part of the design differs from the stated requirements. These data sets are denoted *Check* and *UBR2* respectively.

With the replication model, the input data from the experimental studies [14][15] and the virtual team calculations, the prerequisites for performing the replication study were fulfilled. Four different data sets were available, which enabled four average *iDDR*'s to be calculated. The properties of the data and the construction of the simulation model meant that four simulations runs were carried out. The replication model was set up as described in Table 3.

TABLE 3. Replication simulation setup

Attribute	Value
Number of defects	37; 38
Document size	1
Document complexity	1
Team size	3
Reviewer experience	1
Reviewer expertise	1
iDDR	0.2287; 0.3089; 0.2588; 0.3134
Document coverage	1

The inspected documents contain different number of defects. The corresponding attribute in the simulation model is set according to this for each simulation run. The number of defects, the average *iDDR* and *tDDR* are presented along with the results and analysis in Chapter 5.

4.2.7. Model extension

The extension involved developing an updated version of the simulation model. The purpose of the extension model was to improve the model performance and to better capture the differences between different inspection teams. In particular, the reliance average reviewer performance was not satisfactory.

Instead of using the average *iDDR*, we sample from the set of actual *iDDR*'s from the experiments and choose individual values for each simulation run. This enabled an analysis of the impact on the *tDDR* from the stochastic variation in the *iDDR*. Further, we extended the model to capture different team sizes, and analyze the simulation performance for different team sizes.

4.2.8. Threats

The major threat to the validity of the study results concerns external validity. The replication is conducted on another experiment, hence we cannot conclude whether the model is valid to e.g. industrial inspections. The internal validity threats are under control, as we have got access and good support from the researchers in the original study. We have validated the new implementation of the study, using the data from the original study.

5. REPLICATION RESULTS

5.1. Replication

The results from the replication simulation are briefly summarized in Table 4. The average *iDDR* values are taken from the experiments. The $tDDR_{sim}$ are the outputs from the simulation model. $tDDR_{VT}$ are calculated from the experimental data as a point of reference, and finally, $tDDR_{\Delta}$ is the difference between $tDDR_{sim}$ and $tDDR_{VT}$. The simulation model does fairly well corroborate the results of the experiments.

The average individual effectiveness parameter (*iDDR*) in the four cases varies between 22.9% and 31.3%. The simulation model results in mean team effectiveness (*tDDR*) between 46.9% and 64.2% while the virtual teams, created from the experiment data had an average *tDDR* between 39.9% and 60.8%. The mean deviation between the effectiveness of the simulation and the virtual teams is 3.7%. The deviation is in the magnitude of one standard deviation for three of the four cases.

TABLE 4. Results from the replicated simulation model.

	Average individual defect detection rate - iDDR	**Simulated team defect detection rate - $tDDR_{sim}$**	**Virtual team defect detection rate with standard deviation in parenthesis - $tDDR_{VT}$ (Std dev)**	**Difference in team defect detection rate - $tDDR_{\Delta}$**
Rand	0.2287	0.4687	0.3994 (0.047)	-0.0693
UBR1	0.3089	0.6331	0.5979 (0.066)	-0.0352
Check	0.2588	0.5304	0.5182 (0.13)	-0.0122
UBR2	0.3134	0.6423	0.6083 (0.073)	-0.0340
				Average
				-0.0377

The original study investigated the same range of parameters (iDDR between 14.6% and 32.1% and *tDDR* between 32.2% and 62.9%) and the average deviation between the effectiveness of the virtual teams and the simulation is -1.99%. The model overestimates in both the original and the replicated study.

5.2. Extension

In Table 5 the mean deviation detection rate $tDDR_{\Delta}$ from the simulation run with the extended simulation model is presented. For the chosen team sizes, the general trend is an increasing deviation *tDDR* along with an increasing team size, a trend that seem to apply to all data sets.

TABLE 5. $tDDR_{\Delta}$ for the extended model. The replication results given as a reference.

	Team Size			**Replication (team size 3)**
	3	**4**	**5**	
Rand	0.0696	0.0848	0.0834	-0.0693
UBR1	0.0109	0.0052	-0.0072	-0.0352
Check	-0.0095	-0.0214	-0.0451	-0.0122
UBR2	0.0512	0.0717	0.0773	-0.0340

The *Review Correction Factor* described in Section 4.2.4 limits the contribution to the *tDDR* for an inspection team for each added reviewer's *iDDR*. This means that the factor takes into account that adding more reviewers increases the duplicate defects found. In a real life process, adding more reviewers may not increase the *tDDR* if they find only duplicates. However, the abstraction level of the simulation model is limited and does not reflect this fact, which may contribute to the overestimate of the *tDDR*.

In Table 6 the difference between the variance of simulated and calculated *tDDR* is presented. With an accuracy of two decimals the deviation does not change with an increasing team size. This observation indicates that the simulation model is quite stable for varying number of reviewers.

TABLE 6. Difference between the variance of the simulated and calculated tDDR.

	Team Size		
	3	**4**	**5**
Rand	0.0040	0.0040	0.0040
UBR1	0.0061	0.0064	0.0059
Check	-0.0024	-0.0021	-0.0007
UBR2	0.0059	0.0070	0.0077

6. CONCLUSIONS

Empirical data is scarce and expensive, as it involves measurements on humans working in real-life or experimental processes. Building a process simulation model might help using the collected data more effectively. A study based on this approach is conducted by Armbrust and Münch [11][2]. They managed to build a simulation model of a software design inspection process, that resembles the underlying empirical study rather well, with deviations in the defect detection rate of about 2%.

This paper reports a study, replicating the same study with data from other inspection experiments. Using the same model structure, the replication resulted in a deviation of about 4% in the defect detection rate. Hence we can conclude that the model is useful for other data sets as well.

Still, the model comprises some correction factors that are rather arbitrary chosen, and it does not capture sufficient random variation. The extended simulation model adds a true variation in reviewer performance. The deviations are still in the same range, hence further work includes modifying the model to reduce the reliance on relative factors, and rather build on the experimental data.

For a small and repeatable process such as the inspection process, it seems feasible to develop and use a simulation model. However, it is still a question whether these results can be extended to comprise processes with larger scope and hence variation, e.g. a complete development process.

7. ACKNOWLEDGEMENTS

The authors are thankful to Dr. Jürgen Münch and Mr. Ove Armbrust of Fraunhofer IESE for making the original model and their data available for the replication purpose. The work is partly funded by the Swedish Research Council under grant 622-2004-552 for a senior researcher position in software engineering.

8. REFERENCES

[1] C. Andersson, L. Karlsson, J. Nedstam, M. Höst and B. Nilsson, ”Understanding Software Processes through System Dynamics Simulation: A Case Study”, *Proceedings of the 9th IEEE Conference and Workshop on the Engineering of Computer-Based System*s, pp. 41-48, 2002.

[2] O. Armbrust, Using Empirical Knowledge for Software Process Simulation: A Practical Example, Diploma Thesis, University of Kaiserslautern, Germany, 2003.

[3] A. Aurum, H. Petersson and C. Wohlin, “State-of-the-art: software inspections after 25 years”, *Software Testing, Verification and Reliability*, 12(3):133-154, 2002.

[4] V. R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sorumgard and M. V. Zelkowitz, ”The Empirical Investigation of Perspective-Based Reading”, *Empirical Software Engineering*, 1(2):133-164, 1996

[5] T. Berling, C. Andersson, M. Höst, C. Nyberg, “Adaptation of a Simulation Model Template for Testing to an Industrial Project”, *Proceedings Software Process Simulation Modeling workshop*, Portland, USA, May 3-4, 2003.

[6] S. Biffl and W. Gutjahr, “Influence of team size and defect detection technique on inspection effectiveness”, *Proceedings 7th International Software Metrics Symposium*, pp. 63-75, 2001.

[7] L.C. Briand, K.E. Emam and B.G. Freimut, “A comparison and integration of capture-recapture models and the detection profile method”, *Proceedings of the Ninth International Symposium on Software Reliability Engineering*, pp. 32-41, 1998.

[8] M. Ciolkowski, C. Differding, O. Laitenberger and J. Münch, *Empirical Investigation of Perspective-based Reading: A Replicated Experiment*. Technical Report 048.97/E. Fraunhofer IESE 1997;

[9] B. Efron and R. Tibshirani, “Bootstrap Methods for Standard Errors, Confidence Intervals and Other Measures of Statistical Accuracy”, *Statistical Science*, 1(1):54-75, 1986.

[10] M. I. Kellner, R. J. Madachy and D. M. Raffo, ”Software process simulation modeling: Why? What? How?” *Journal of Systems and Software*, 46(2-3):91-105, 1999.

[11] J. Münch and O. Armbrust, “Using Empirical Knowledge from Replicated Experiments for Software Process Simulation: A Practical Example”, *Proceedings of the International Symposium on Empirical Software Engineering*, pp. 18-27, 2003.

[12] D. M. Raffo and M. I. Kellner, ”Empirical analysis in software process simulation modelling”, *Journal of Systems and Software*, 53(1):31-41, 2000.

[13] C. Robson, *Real World Research*, Blackwell publishers, 2002.

[14] T. Thelin, P. Runeson, and B. Regnell, “Usage-Based Reading - An Experiment to Guide Reviewers with Use Cases”, *Information and Software Technology*, 43(15):925-938, 2001.

[15] T. Thelin, P. Runeson and C. Wohlin, “An Experimental Comparison of Usage-Based and Checklist-Based Reading”, *IEEE Transactions on Software Engineering*, 29(8):687-704, 2003.