An empirically validated simulation for understanding the relationship between process conformance and technology skills

Matalonga, Santiago; Solari, Martín; San Feliu, Tomás

Published in:
Software Quality Journal

DOI:
10.1007/s11219-013-9214-2

Published: 01/12/2014

Document Version
Peer reviewed version

An empirically validated simulation for understanding the relationship between process conformance and technology skills

Santiago Matalonga – Martín Solari - Tomás San Feliu

Abstract: Software development is a fast-paced environment where developers need constant update to ever changing technologies. Furthermore, process improvement initiatives have been proven useful in increasing the productivity of a software organization. As such, these organization needs to decide where to invest their training budget in. As a result, training in technological update to their workforce or training in process conformance with its productive processes becomes conflicting alternatives.

This paper presents a systemic simulation of a production environment. The objective of this simulation is to understand the changes in behavior when selecting either one of the above training alternative. The simulation has been empirically validated – using statistical process control - against the performance baseline of a real software development organization. With the simulation under statistical control and performing like the baseline, the independent variables representing process conformance (process training) and technology skills (skills training), were modified to study their impact on products defects, and process stability. Our results show that, while both variables have positive impact on defects and process stability, investment in process training results process with less variation and with fewer defects.

Keywords: simulation; software process improvement; statistical process control

1 Introduction

The software development industry is a fast-paced environment with continuing changing technological cycles and ever demanding knowledge requirements for its workforce. For instance, it has been over a decade since the introduction of the .NET framework, and it has already undergone four major revisions. This is an example the need of constant training in order for developers to continue to be competitive in the marketplace.

On the other hand, software process improvement models like ISO 9001:2008 (ISO, 2008) and CMMI (CMMI Product Team, 2010), have grown in popularity and adoption in the recent years (SEI, 2011). Among other things, these models share the vision that training is an effective tool to increase developer's productivity by training them to perform as a described in the organization's standard process.

This paper addresses the question of where to place the training investment for increasing the benefits.

This paper presents a systems dynamic model designed to study the impact on training in a software development process. This model will study the relationship between process conformance (represented by the process training variable) and technological update (represented by the skill training variable). The model will show how a stable process reacts when training is incremented and how these variables affect product variables like non conformances and defects. According to statistical process control, a process is considered stable when only natural causes of variation are present in its output (Florac, Park, & Carleton, 1997).

By taking advantage of data available from a software factory, the simulation was first empirically calibrated to reproduce the behaviour of a software development organization. The calibration was achieved with the use of data from live development projects. By tampering with the independent variables we were able to observe how the behaviour of the system changes. Our results show that, training has more impact if it is placed on process conformance than in technological update.

This paper is organized as follows; first we describe in section2.1 the research available into the use of simulation in software engineering research. Section 2.3 presents a short overview of the statistical process control tools that will be used to analyze the model's output.

In section 3 the model is described in detail. Section 3.1 describes the concepts that have been modelled, while section 3.2 presents the rationale for including and excluding elements of the observer reality. Finally section 3.4 describes the validation test to which the models' development was subject to.

Section 4 , presents the results of the model execution, and our analysis is presented in section 5 .

Finally, threats to validity of our results are described in section 6 before finishing the paper with our conclusions.

2 Background

2.1 Simulation and system dynamics in software engineering research

System dynamics was pioneered by the work of Forrester at MIT (Forrester, 1969). It is a discipline which is well suited to address problems in dynamic environments (Senge, 2006) where:

- The system is dynamic as opposed to static.
- It is driven by a mindset of thinking in circles, or loop-causality.
- The system is the cause rather than the effect.

Software production processes are fit for applying systems thinking, since they are highly volatile, non-linear systems (Anderson, 2003; Pfahl & Lebsanft, 1999). In (Kellner, 1999) the authors present a systematic review of the application of system dynamics in the software engineering domain, including examples of studies on: software process improvement; productivity; software quality assurance.

One of the earliest works on system dynamics applied to software engineering is the work by (Abdel-Hamid & Madnick, 1991). In their book, these authors present applications of systems model to software project management. The work by (Madachy, 2008), focused on modeling the dynamics of a software production

process, and to provide a way to enable practitioners to acquire practical knowledge about the dynamics of their software process by studying, reproducing and enhancing the models provided in his book.

Furthermore, there has recently been increased interest in using simulation models, among them system dynamics models, as theory building constructs. For instance (Cocco, Mannaro, Concas, & Marchesi, 2011), developed a systems dynamics model to compare traditional vs. agile practices. Furthermore, the empirical software engineering community has also identified simulation models, in general, as a cost effective alternative to experiments with human subjects (Barros, Werner, & Travassos, 2000).

To the best of our knowledge, there have been no attempts at using a systems dynamic model to study the impact of training on defects and process stability. Specifically our research question is: How does a stable process reacts when process conformance or technological skills have been improved by training?

2.2 Elements of a systemic simulation

Systemic simulations try to answer how a model behaves through time. These models are usually represented by a set of differential equations, which represent the how the quantities being modeled evolve through time (Müller & Pfahl, 2008). There are three basic information items that are used to create a systems model: stocks, flows, and information links. Stocks are repositories of products, or information. For instance, Fig 1 presents a stock containing software defects. Usually, the degree to which a stock is "painted" represents how full that stock is in the simulation.

Fig 1 Example of a stock

Flows interconnect stocks or the outside world (represented by the cloud in Fig 2) of the simulation with another stock. A flow establishes the rate in which the stock is filled or emptied. Fig 2 represents how coding injects defects into the product defect stocks.

Fig 2 Example of a flow

Finally, information links are non physical relationships that occur between flows or stock and flows. For instance, Fig 3 shows that the number of products defects impacts the probability of finding them by a testing team. By convention, names of flows are gerunds, in order to indicate that an action is taking place.

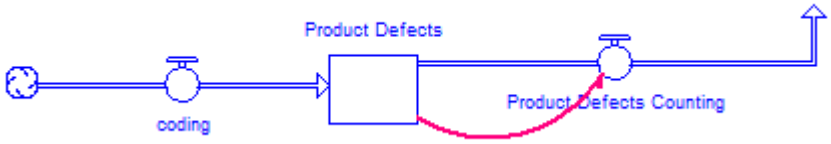

Fig 3 Example of an information link

Although there are others shapes that can be used within a systems dynamic model(Madachy, 2008), these are the only ones that were used within the model presented in this paper.

2.3 Evaluating process capability

Statistical Process Control (SPC) is an economical way of managing a process output (Shewhart, 1980). The idea is to obtain measures of a process' output in order to make sure that it operates within its specification levels.

Process capability is defined as the ability of the process to produce an output within the specification limit (Pyzdek, 2003). The process capability indexes determine the goodness of a process output when compared to its specification limits. In this work we will use the following process capability index to compare the results of the different simulation runs.

$$C_{pk} = \min\left(\frac{USL - \bar{x}}{3\sigma}, \frac{\bar{x} - LSL}{3\sigma}\right)$$ Where:

- USL represents the upper specification limit
- LSL represents the lower specification limit
- $\bar{x}$represents the mean
- σ Represents the standard deviation of the sample.

The higher the C_{pk}, the most likely the process output is to fulfill its specification goals (Pyzdek, 2003).The C_{pk} index will be used to determine if process capability improves when compared to the empirical dataset.

Furthermore, run charts will be used to visually evaluate if different runs of the simulation changes the output. We will observe if there is a shift in the control limits of the run charts take that as a signal of a change in behavior of the process (Florac et al., 1997).

3 Design and development of the simulation model

3.1 Software engineering concepts included in the model

As we have mentioned, the intention of the model is to understand the interactions between defects and process conformance. In this section we will define these concepts:

- Process conformance

According to the ISO 9000 standard, conformance is understood as deviations from the quality plan - clause 4.17 (ISO, 2008). Nevertheless, process conformance is not formally defined, apart from mentioning the importance of the correct execution of the defined processes and approved operational standards. ISO 19011, which is a guideline for performing audits on a quality management system, defines a non-conformance as "the non-fulfilment of a specified requirement"(ISO, 2000). Likewise, the CMMI model does not explicitly define the term non-conformance. Nevertheless, it does mention that process non-conformances are the typical output of an objective audit into the organization's processes.

In this work, we will define "non conformance" as an individual incident that has been raised during a process audit – a deviation from what has been defined in the standard process. It will be the variable used to measure process training in the simulation.

- Software defects

Software engineering literature has dealt with the definition of the term "defect". The IEEE 610.12-1990 standard makes a comprehensive distinction between

defect, error and fault (IEEE, 1990). Six Sigma provides a vision of defect from the perspective of the customer (Pyzdek, 2003).

In this work we will take a more general definition of a defect. We will use the definition that is used in the software factory were we have conducted our study. This is, a defect is defined as an inconsistency between software requirements and the software under execution. Defects are detected by an independent functional testing team.

- Relationship between non-conformances and software defects.

Understanding this relationship is one of the contributions of the model, but at the same time one of the main design decisions taken.

First of all, a characteristic of non conformances is that they can happen at any point of the lifecycle of the projects. Nevertheless, they will go unnoticed until they are revealed by process audits.

When team members execute an organization's process in order to build software they are bound to skip or short cut the process at some point during the lifecycle. And, unless training has been specifically put in place to prevent this, the team member will continue to short cut the process until an audit reveals the short cut, or he understands the need for the process through training (Pfahl, Klemm, & Ruhe, 2001).

In CMMI, investment in process audits or process conformance is supported by the idea that if processes and business needs are aligned, those projects who follow the defined process should commit fewer defects. In this work we will take this assumption as a valid premise.

3.2 Design requirements

An essential model requirement is that it should provide an accurate and accepted representation of the reality under study. In this case, we are modeling the software development process of a software organization – a software factory that specializes in .NET technologies.

Furthermore, it is a design requirement that the model mimics the reality under study both in structure and behavior.

To validate the first restriction, a set of qualitative questions will tested against the model to assess is depiction of the reality (see section 3.4.1).

In addition to this, in its initial state, the model should *behave as the empirical dataset* - taken from the same software factory (see Table 1). In order to determine this, we will apply the following SPC criteria: the model in its initial execution should produce output that lies within the control limits of the data set.

Table 1 Empirical data set

Project	Number of Non Conformances	Product Size	Total Defects
Project 1	8	354	186
Project 2	11	296	125
Project 3	16	918	431
Project 4	13	722	174
Project 5	9	749	216

3.3 Design rationale

This section describes the model and the decisions that drove its design. Table 2, presents a qualitative description of the main characteristics of the model.

Table 2Qualitative description of the model

Scope	The model will represent an instance of a project
Audience	Senior management, training managers and process improvement groups within software development organizations.
Required knowledge	- Basics of process improvement and knowledge of process improvement models. In our case we are using CMMI terminology. - Systems modeling basics, including understanding of the concepts of stock, rate, and information link.
Research question	Is it better to invest training effort on technology skills or on process compliance?
Variables under study	skills training and process training.
Model variables	Software defects, project non conformances, product size.

The model is presented in Fig 4. Following the advice by (Pfahl & Lebsanft, 1999) we will discuss the main design decisions taken during the model development process.

The model is vertically divided in three distinctive areas:

- The non-conformances area (top swim lane).
- The defect accumulation area (middle swim lane).
- The product size area (lower swim lane).

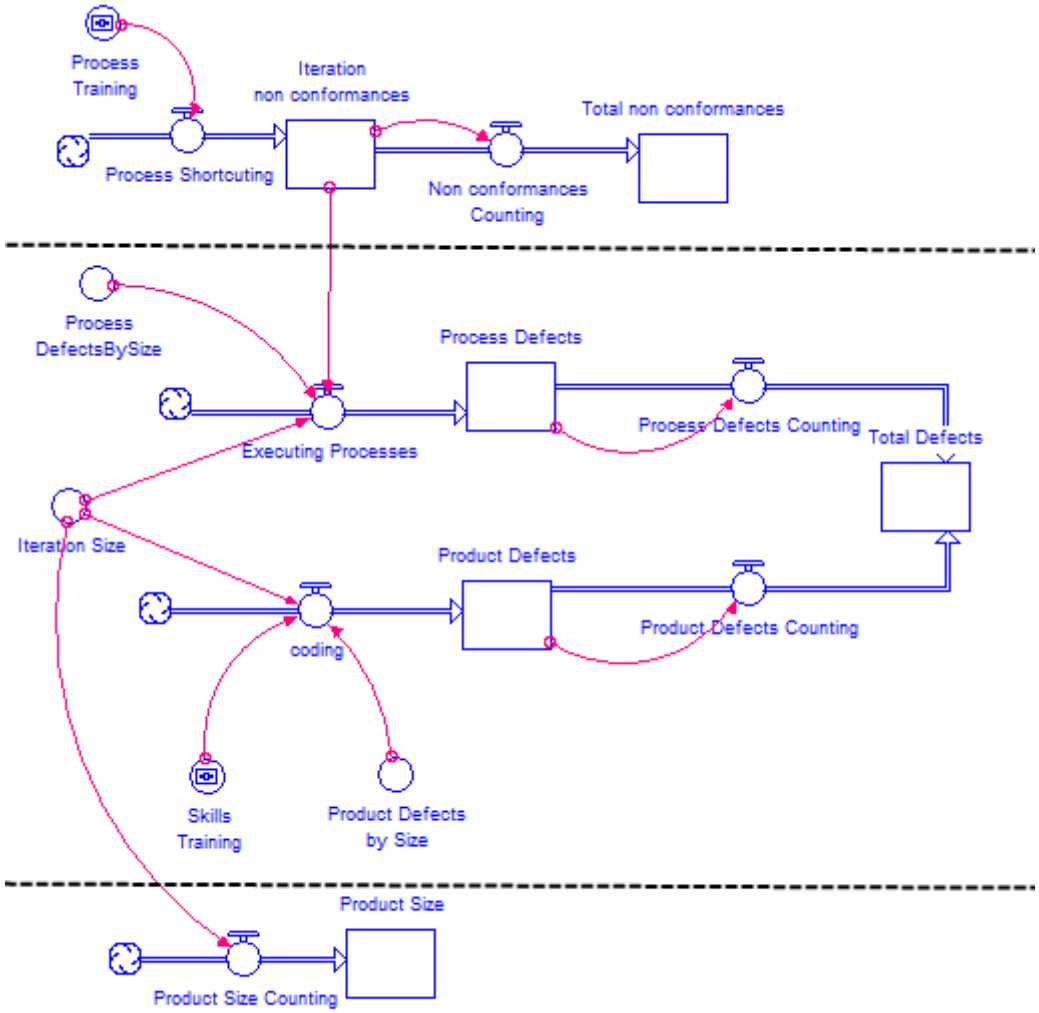

Fig 4 Systems model

The top swim line of Fig 4 presents the non-conformances area. As we mentioned in the previous section, either if team members are deliberately short cutting the process, or if they have honestly skipped it, non conformances will not be detected until an audit has been executed. This reasoning lead to the following assumption: there is a delay in the feedback cycles for non conformance. They depend on when the audit has been scheduled. To accommodate this in the simulation, each iteration of the simulation will represent and individual project. Feedback on non-conformances is visible on a project by project timeline. This

will be contempleted in the simulation as we modify the independent variables (see Table 3).

The middle swim lane of Fig 4shows the product defect area. Product defects are divided in two flows: those who have been influenced by process defects and those that arise from software engineering activities. This is a distinction that has its roots on how the organization classifies non conformances; they have designed a taxonomy that assigns each non conformance to a CMMI process area (CMMI Product Team, 2010).

The lower part represents the software production process (requirements, design, code and test). To balance simplicity and maintain the focus in the research question we decided to make no distinction into where in the production process the defect is injected. It could have been feasible to take this into account in our model, since it has already been modeled in (Madachy, 2008), but it was deemed unnecessary since because modeling the production process is not focus of our research question. Likewise, it was also decided not to include the work feedback cycle within this model. Nevertheless, in (Matalonga & San Feliu, 2009) we presented a systems model to understand the effects on defect classification, detection and rework for this same software factory.

The upper part represents the defects that arise from the execution of the software process. This flow represents process non-conformances that the process has transformed into defects in the software product.

Finally, the lower swim line of Fig 4 shows the stock that is used to count the total size of the project. The *Product Size Flow* takes input from the *Iteration Size variable* which has been adjusted to produce outputs according to the empirical data set.

The last important item contemplated in the model is the relationship between defects and non-conformances. This is represented by the information link between the *Iteration non conformances* stock and the *Executing Processes* rate.

During a previous observation of the same software factory we statistically demonstrated that non-conformances and process related defects were positively correlated (Matalonga & San Feliu, 2010). With the data presented in Table 1, the Pearsons' correlation coefficient, between *Total non conformances* and *Total Defects*, is significant at 0.88. Our interpretation is that non-conformances act like an amplifier of production defects. Therefore, the more non-conformances a

project commits, the more defects a project injects. The direction from non conformances to defects is result of the assumption that processes and business are aligned.

Finally, Table 3 presents the definition of the variables included in the model. The leftmost column indicates which variables will be modified in the simulation to study the behavior under different values.

Table 3 Description of variables included in the model

Variable	Definition	Modified in the scenarios
Process Training	A measure of resources skills while executing the process of the project. It is predefined before the execution of the scenario.	Yes
Process Defects by Size	Process defects injected by size of the iteration. First estimated from the empirical data set, and adjusted during model development	No
Iteration Size	A Normal distributed variable representing the size of the iteration. Size is measured in an the organization's size unit - loosely based on function points(Albrecht, 1983).	No
Skills Training	A measure of resources skills in the development technology for the project. It is predefined before the execution of the scenario.	Yes
Product defects by Size	Process defects injected by size of the iteration. First estimated from the Empirical data set, and adjusted during model development	No

3.4 Model validation

We have qualitatively and quantitatively validated the model. For a qualitative evaluation, we have analyzed the dimensions presented by (Schwaninger & Grösser, 2008). As for quantitative evaluation, we have tuned the model variables so that the model behaves as the observed software factory.

3.4.1 Qualitative validation

In this section we present a validation of our model in terms of 12 dimensions proposed by (Schwaninger & Grösser, 2008).

Refutability; refers to the ability of a theory to be refuted or falsified. In this aspect we present a systems model which has been executed, and the main parameters are also presented. Therefore, it is possible for any third party to reproduce the model and propose defects or improvements to it.

Importance; refers to the significance or acceptance of the theory by stakeholders within the models domain. In our case, the impact that training and process improvement have had in the industry, are the basis for our claims of the importance of the question under study.

Precision and clarity; refers to the hypothesis of the model being clear and easily developed from the model. Through the design of the model, we have been careful at abiding by the definition of the concepts included in the model. Moreover in this paper we have paid close attention to communicating our assumptions when establishing the relationships within the model.

Simplicity; refers to a limitation of complexity and assumptions. Our working hypotheses have been made clear in section 3.2 . And, even though it could have been possible to include other aspects in the model, we valued simplicity over comprehensiveness. Our intention was to understand the impacts of the variables under study and not to produce a precise simulation of the observer reality.

Comprehensiveness; refers to the completeness of the coverage of the substantive areas of interest of the model. As mentioned in "simplicity" the model is comprehensive enough in its ability to answer the question, but other aspects and interactions which can be related to the research question could have been included in the model.

Operationality; refers to the model being testable and measure. This is positively affirmed since we have designed and implemented the model in a simulation package.

Validity; refers to the model being an accurate representation of reality. Our quantitative validation against observed data in a real production setting, gives enough confidence that the model behaves as the empirical data set. Nevertheless as it was explained with the "simplicity" concept there are several other variables that might have influence in the outcomes under study that were explicitly left out.

Reliability; refers to the model being free of measurement errors. Our intention is that the model provides insight into the improvement of the different training options in light of the variation present in the production environment. Since the

model's variable definitions respect the definition of the software factory. And the model behaves as the empirical data set, it is reasonable to think that it is also subject to the same types of error present in the data set.

Fruitfulness; refers to the usefulness of the statements that can be made out of the model. The design and subsequent execution of the model has leaded us to the following affirmations (see the following sections):

Investing in process training reduces variation measured in non conformances and, if process and business are aligned, this variation has the overall effect of reducing defects.

Investing in engineering skills training also reduced overall production defects, but process variation does not decrease accordingly.

Practicability; the model provides a conceptual framework for practice. This characteristic could have been evaluated better if we had been successful at taking decisions in the software factory after the presentation of the model. Unfortunately, their attention shifted and an evaluation of the model from the practical viewpoint of a development organization could not be achieved.

This section showed that the model is an adequate representation of the reality. Threads to validity arise from design decisions which were consciously taken; this will be discussed further in section.

3.4.2 Quantitative validation

The initial values were the slope and the Y-intercept of the linear regression line for "Total non conformances" and "Total defects". These values were tuned until the model produced outputs (defects, non-conformances and project size) that could be comparable to the empirical data set.

Finally, in order to introduce variation, all model variables were initialized with the Normal distribution function of the IThink software package[1], with the average taken from the empirical data set, and three times standard deviation (to allow for visible variations with ten iterations of each scenario). We completed several runs of the model to confirm that the exhibit behavior always produced outputs within the control lines of the empirical data set (the last 10 runs where included in this paper and are presented as Scenario 1, see section 4).

[1] http://www.iseesystems.com/

4 Model execution scenarios

In order to test our research question, we have designed three scenarios to which the model will be executed. A scenario is defined as a set of 10 executions of the simulation with the *same values* for the independent variables. We have defined three scenarios to test the model: *Scenario 1: Baseline*; *Scenario 2: Process Training* and *Scenario 3: Skills training*.. Scenarios 2 and 3 differed for scenario 1, in that the variables that represent training are tampered in order to simulate that training has given to the resources performing the work.

Scenario 1: Performance Baseline. The objective of these runs is to adjust the model parameters so that it behaves as the observed empirical data set. Once enough confidence is gained with the behavior of the model the training variables can be adjusted to study the behavior of the system.

The scenario 1 will be considered a successful representation of the Empirical data set if: (1) there is no significant shift in the process control lines for each dimension (non-conformance; product size, total defects); (2) the C_{pk} index for the Scenario 1 are lower than in the Empirical dataset.

Since is that the simulation has more data points than the empirical data set, the individual run chart will more descriptive power for the scenarios (Florac et al., 1997).This means that it is expected that the control limits should shrink around the average. Therefore, we will perform the Student's t test of means to study if there is a significant shift in the averages between the empirical data set and scenario 1.

Scenario 2: Process training. This scenario represents the state in which human resources have been thoroughly trained in the organizations process and they have been given no training to improve their skills in the technology. The process training variable will be increased fourfold and the skills training variable will be left with the nominal values.

Scenario 3: Skills training. This scenario represents the state in which human resources have been thoroughly trained in the development technology and they have been given no training to improve their knowledge of the process. The skills training variable will be increased fourfold and process training variable will be left with the nominal values.

4.1 Evaluation objectives

We will analyze the three scenarios and their behavior against each other and against the empirical data set (see Table 1). As mentioned, the objective of this study is to evaluate how training affects the performance of a software process. This objective will be broken down in the following questions:

I. Does scenario 1 behave as the empirical data set?

II. How does the outputs of scenario 2 and 3 compare against each other?

III. How do they compare against scenario 1?

Each question will be analyzed by observing the shifts in the process control limit and the evaluation of the process capability index. As a reference, the control limit for the empirical data set will be used as the specification limits in the application of the C_{pk} formula for the scenarios.

4.2 Process simulation runs

Table 4presents the results of the execution of the scenarios, numbered columns correspond to a simulation run within one scenario.

Table 4 Process simulation results

Scenario 1	**Run**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	Process training	x1									
	Developers training	x1									
	Non-conformances	10	11	8	10	11	8	7	8	12	10
	Size	480	683	395	551	492	679	529	712	746	748
	Defects	150	219	177	263	168	285	145	347	216	262
Scenario 2	**Run**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
	Process training	x4									
	Developers training	x1									
	Non-conformances	2	2	2	2	2	1	3	2	2	2
	Size	654	562	439	709	652	527	758	579	454	673
	Defects	199	173	141	143	159	133	213	188	158	197
Scenario 3	**Run**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
	Process training	x1									
	Developers training	x4									
	Non-conformances	7	10	7	8	11	8	10	9	8	7

	Size	707	608	550	565	703	671	588	631	571	639
	Defects	294	211	187	229	266	196	201	235	165	153

With the previous data, control limits were calculated for each of the scenario. Fig 5 presents a process control graph with all scenarios included. The vertical swim lines represent the division between the scenarios.

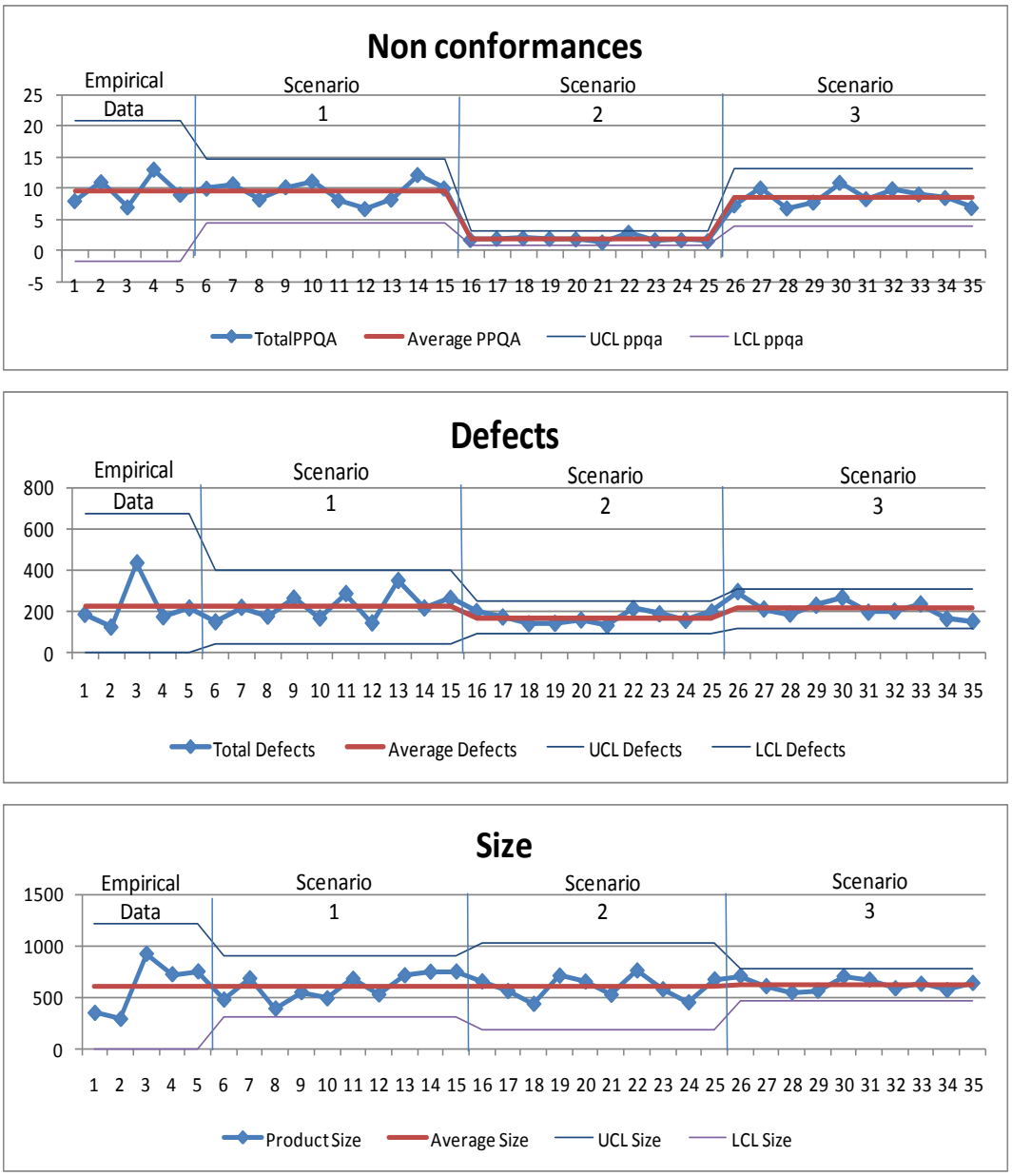

Fig 5 Process execution runs

The following table presents the C_{PK} indexes for each of the process dimensions associated to the process output measures (non conformances and defects). The higher the C_{PK} the better the result of the scenario is.

Table 5 Cpk by scenario

Simulation Run	Non conformances (C_{PK})	Defects (C_{PK})
Empirical Data Set	1.6	0.6
Scenario 1	2.3	1.1
Scenario 2	3.0	9.6
Scenario 3	2.5	2.8

5 Analysis

This section will present a discussion of the results guided by the questions presented in section 4.1 .

I. Does scenario 1 behave as the empirical data set?

The first task was to validate the model by assuring that the scenario 1 behaves like the Empirical data set. This is an important step in order to demonstrate that our design criteria have been met and that the model is a faithful representation of the reality. In Fig 5 the comparison between the averages and the control limits of the empirical data set and scenario 1 can be observed. A root cause that explains the observed reduction in the control limits is the fact the simulation scenarios have more data points than the empirical data set.

Furthermore, a Student's t test to determine whether there was a difference within the averages of the process variables (size, non-conformances, defects) between the empirical data set and scenario 1. This revealed that there are no significant differences among the average of the process variables in the scenario 1, so the difference can be attributed to chance. The results are presented in Table 6.

Table 6t-test for equality of means for scenario 1

Process Variable	**Levenes's test for equality of means**		**t-test for equality of means**		
	F	Sig	t	Df	Sig (2 tailed)
Non conformances	1.212	0.291	0.08	5.9	0.94
Size	10.522	0.06	0.05	4.9	0.96
Defects	1.066	0.321	0.06	5.2	0.96

This result gives enough confidence that the model behaves like the empirical data set, and therefore the initial conditions can be modified to study how the system behaves under different conditions.

II. How does the outputs of scenario 2 and 3 compare against each other?

Fig 5 shows that scenario 2 presents the smallest average of defects and also the smallest variation. Therefore, not only has the process in scenario 2 decreased the number of defects on the organization projects, but it has also limited variation. This makes for more predictable processes. Therefore, a project executing under scenario 2, produces fewer defects and exhibit more controllable processes. Moreover, C_{pk}, comparisons also show that scenario 2 outperforms scenario 3.

A final note is that in scenario 2, there were significant differences in the size variable. Taking into account the model design, the size variable has no input from any other model variable; so we will disregard this observation and understand it as a symptom of the diffusion principle of a random variable (Feller, 1968).

III. How do scenario 2 and 3 compare against scenario 1?

Both scenario 2 and 3 outperform scenario 1 in terms of reduced defects, non conformances and process variation. Hence our model confirms the impact of training in the performance of the workforce. Furthermore, scenario 2 accounts for fewer defects, and less variation. This is an asset for production environments. Therefore, according to our model and simulations, investing training in process conformance has more impact than investing it in technology skills. We can assert this because Scenario 2 produces less defects and a more controllable process, and because it is the scenario with the highest the $C_{pk}$for each of the two output variables.

6 Threats to validity

This section presents a discussion on the threats to validity that can influence the conclusions on this paper. This discussion will be based on the categories of threads to validity described by (Wohlin et al., 2000).

Conclusion validity is concerned with the relationship between the treatment and the outcome. There must be statistical significance to support the relationship. The simulation is inherently quantitative; furthermore, the conclusions expressed have been based in statistical process control guidelines which build enough confidence ($p > 95\%$) to support the conclusions taken.

Internal validity is concerned with making sure that the relationship observed between the treatments is causal and that no outside factor can have an influence on the outcome. In the case of a simulation, once the scope and variables have

been set, no other variables within the simulation can affect the outcome. In this case, the simulation can only be affected by the software of the simulation platform in which the model was developed.

External validity is concerned with generalisation of the results. External validity focuses on whether the conclusions can be statistically inferred from the data.Whoever tries to use the model should first evaluate if the variables used in the model represents an accurate description of their reality. In this work the model has been designed against a single software factory in Montevideo, Uruguay.

In addition to this, they should also make sure that their observed reality behaves like the empirical data set. If it is not, it would be best to re-calibrate the model until it has obtained results that more accurately represent the new reality.

Finally, construct validity is concerned with the degree at which the actual measurements correlate to the theorized measurements. In the case of this study, construct validity is highly influenced by the decisions taken during the development of the simulation – for instance the definition of a defect used in the model. In order to minimize this risk we have taken the following actions. In section 3.2 we have conveyed the decisions taken during the development of the model. Furthermore, in order to validate it, we had set quantitative goals for the model to behave as the observed software factory. And in section 3.4 we have presented a qualitative validation of the model.

7 Conclusions

This paper has presented a systemic simulation which was developed in order to understand how a stable process behaves when process conformance or technological skills of the workforce is improved. The model presented was quantitatively and qualitatively validated. First, it was built to simulate the behaviour of a software organization. A baseline execution of the model has been presented which shows that its' outputs perform at levels within the variation of the software factory. Furthermore, qualitative evaluation of the model was carried out in order to minimize the researchers' bias at modelling the reality.

The model variables were then adjusted to study the research question in this paper. Three scenarios were designed to evaluate the research question. The

resulting execution shows that investing training in process conformance results in more predictable processes and a product with fewer defects.

Finally, it has not been modelled how outdated technology skills impacts on quality in the long run. It is reasonable to expect that as the gap widens other factors will start limiting the effectives presented in this paper (a patter know as "limits to growth" in systemic thinking). Therefore, we suggest that organizations looking to take benefits from our results should establish measurements to control the results of either type of training.

8 References

Abdel-Hamid, T., & Madnick, S. (1991). *Software Project Dynamics: An Integrated Approach*. Prentice Hall PTR.

Albrecht, A. J. (1983). Software function, source licnes of code, and development effort prediction: A software science validation. *IEEE Transactions on Software Engineering*, *9*(6), 639-648.

Anderson, D. (2003). *Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results*. (P. Coad, Ed.)*Coad Series on Software Engineering*. Prentice Hall PTR.

Barros, M. O., Werner, C. M. L., & Travassos, G. H. (2000). Using process modeling and dynamic simulation to support software process quality management. *XIV Simpósio Brasileiro de Engenharia de Software, Workshop de Qualidade de Softwarw*. João Pessoa: SBC.

CMMI Product Team. (2010). CMMI for Development, Version 1.3. Improving processes for developing better products and services. Software Engineering Institute.

Cocco, L., Mannaro, K., Concas, G., & Marchesi, M. (2011). Simulating Kanban and Scrum vs. Waterfall with System Dynamics. *12th International Conference Agile Processes in Software Engineering and Extreme Programming*. Madrid 2011. doi:10.1007/978-3-642-20677-1_9

Feller, W. (1968). *An Introduction to the Probability Theory and its Applications*. New York: John Wiley & Sons.

Florac, W. A., Park, R. E., & Carleton, A. (1997). Practical Software Measurement: Measuring for Process Management and Improvement. Pittsburgh: Software Engineering Institute.

Forrester, J. W. (1969). *Urban dynamics*. Waltham, MA: Pegasus Communications.

IEEE. (1990). IEEE STD 610.12-1990. Standard Glossary of Software Engineering Terminology. (Institute of Electrical and Electronics Engineers, Ed.)*IEEE STD 610.12-1990*.

ISO. (2000). ISO 19011 - Quality Management System Auditing. (International Standards Organization, Ed.).

ISO. (2008). ISO 9001:2008(E). Quality Management Systems - Requirements. (International Standards Organization, Ed.).

Kellner, M. (1999). Software process simulation modeling: Why? What? How? *Journal of Systems and Software*, *46*(2-3), 91-105. doi:10.1016/S0164-1212(99)00003-5

Madachy, R. J. (2008). *Software Process Dynamics*. Wiley-IEEE Press.

Matalonga, S., & San Feliu, T. (2009). Using defects to account for the investment in support process areas. *Europe SEPG 09*. Prague, Czech Republic.

Matalonga, S., & San Feliu, T. (2010). Using Process and Product Quality Assurance Audits to Measure Process Business Alignment. *International Conference of Software and Systems Engineering and their Applications*. Paris, France.

Müller, M., & Pfahl, D. (2008). Simulation Methods. In F. Shull, J. Singer, & D. I. K. Sjoberg (Eds.), *Guide to Advances Empirical Software Engineering*. Lysaker, Norway: Springer.

Pfahl, D., Klemm, M., & Ruhe, G. (2001). A CBT module with integrated simulation component for software project management education and training. *Journal of Systems and Software*, *59*(3), 283-298. Retrieved from http://www.sciencedirect.com/science/article/B6V0N-44C84K6-7/2/57eb13c7610fc32fe2a44125c04b7207

Pfahl, D., & Lebsanft, K. (1999). Integration of system dynamics modelling with descriptive process modelling and goal-oriented measurement. *Journal of Systems and Software*, *46*(2-3), 135-150. Retrieved from http://www.sciencedirect.com/science/article/B6V0N-3WN6V65-6/2/0865484a773044b2d091a0183a15f7bf

Pyzdek, T. (2003). *The Six Sigma Handbook: The Complete Guide for Greenbelts, Blackbelts, and Managers at All Levels*. (McGraw-Hill, Ed.) (Second Edi.). McGraw-Hill.

SEI. (2011). CMMI® for SCAMPI Class A Appraisal Results 2011 Mid-Year Update. (S. E. Institute, Ed.)*CMMI Maturity Profile*. Pittsburgh: Carnegie Carnegie Mellon University Mellon University. Retrieved from http://www.sei.cmu.edu/cmmi/casestudies/profiles/pdfs/upload/2011SeptCMMI-2.pdf

Schwaninger, M., & Grösser, S. (2008). System Dynamics as Model-Based Theory Building. *Systems Research and Behavioral Science*, *25*, 447-465. doi:10.1002/sres.914

Senge, P. (2006). *The Fifth Discipline: The Art & Practice of The Learning Organization* (1st ed.). Broadway Books.

Shewhart, W. (1980). *Economic Control of Quality of Manufactured Product*. American Society for Quality.

Wohlin, C., Höst, M., Runeson, P., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2000). *Experimentation in software engineering: an introduction*. Norwell, Massachusetts: Kluwer Academic Publishers.