

This document is published at:

Dugarte, G., Sánchez, M.I., Amescua, A. Medina, F., Armenia, S. (2021). Using system dynamics to teach about dependencies, correlation and systemic thinking on the software process workflows. *IET Software*, 15(6), pp. 351-364

DOI: [10.1049/sfw2.12031](https://doi.org/10.1049/sfw2.12031)


© 2021, The Authors. IET Software published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.



This work is licensed under a [Creative Commons Attribution-NonCommercial](https://creativecommons.org/licenses/by-nc/4.0/)

## ORIGINAL RESEARCH PAPER

# Using system dynamics to teach about dependencies, correlation and systemic thinking on the software process workflows

German-Lenin Dugarte-Peña<sup>1</sup>  | María-Isabel Sánchez-Segura<sup>1</sup> |  
Antonio de Amescua<sup>1</sup> | Fuensanta Medina-Domínguez<sup>1</sup> | Stefano Armenia<sup>2</sup>

<sup>1</sup>Computer Science and Engineering Department,  
Universidad Carlos III de Madrid, Madrid, Spain

<sup>2</sup>Link Campus University, Rome, Italy

## Correspondence

German-Lenin Dugarte-Peña, Av. De la  
Universidad, 30, 28913, Leganés, Madrid, Spain.  
Email: [gdugarte@inf.uc3m.es](mailto:gdugarte@inf.uc3m.es)

## Funding information

Comunidad de Madrid, Grant/Award Number:  
EPUC3M17 - V PRICIT

## Abstract

It is important to count on tools to help software professionals to evaluate the software process and how it may be affected by factors related to its deployment. Simulation models are a valuable means to illustrate the behaviour of such a process since scenario generation supports the prediction of potential outcomes and the prevention of undesired scenarios which are harmful to the process and the company in charge of the project to be developed. This work explores the effectiveness of introducing system dynamics (SD) models in the software engineers' process of understanding, from a management perspective, the software process dynamics. The used SD simulation model of the software process emphasises the representation of an iterative process. The COCOMO II model drivers and their main attributes were used, providing a set of reference factors that affect the software process, the estimation of project cost and the effort required. A set of 59 junior software professionals with no previous knowledge about SD participated in a validation study. For simple predictive scenarios, there was no important improvement effect, while for more complex predictive scenarios SD helped them to guess better and provide a rationale for the expected behaviour of the software process performance.

## 1 | INTRODUCTION

The modelling and simulation paradigms have been oriented towards industrial, chemical, tangible and measurable processes, whose parameters are under the absolute control of those involved in the process. However, in recent years there has been some evolution in the spectrum of approaches, motivated by the need for controlling aspects such as the strategic management, the search for process improvements, or training in terms of software project management.

A potential impact, which motivates the pursuit of advances in the modelling and simulation of software processes, is the reduction of the understanding gap between the software project team and the people interested or involved as beneficiaries of the product to be developed. A software process simulation tool can facilitate a common language that both developers and *stakeholders* can understand, setting an

interesting dialog around the process. This is of interest to project managers and others in software project roles, who have strong technical language tools such as UML, used to abstract and represent their system to be developed. Such complex language can be confusing for *stakeholders* who tend to master a much less technical language closer to the business world and may lose interaction and sufficient understanding of the domain of the problem, if it is addressed in conjunction with and driven by the development team [1, 2].

To deepen the knowledge in this context, it is possible to mention two concrete knowledge bases that already have numerous research works and concrete projects and developments related to this research approach. First, there is a structured conceptual framework for modelling and simulating systems in general, with multiple applications and uses in areas and industries such as chemistry, electromechanics etc. Second, there is a conceptual basis

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.

© 2021 The Authors. *IET Software* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.

around the understanding of software development processes; this allows us to explore different development models from the most classic ones, such as the Waterfall-structured model, to more organic and less rigid models, such as the Rational Unified Process (RUP), or other iterative models like the Craig Larman model [3] that will be taken as a reference in this work, which is explained ahead and also represented in Figure A1 available in Annex A.

From the two knowledge bases mentioned above, a research interest arises around the modelling and simulation of software processes, a field that has been little explored until now and in which the diversity and dynamism of the intervening factors make it so complex that they notably affect the behaviour of the development teams and therefore all the organisational behaviour. The growing boom in software development, the increasingly complex demands on the capabilities of software products, and the growing desire on the part of the industry and the academia to understand and bring software products closer to the reality they serve, make it an important need to have tools that allow a better understanding of these systems and their complexity with the least possible consumption of resources. Software development processes' modelling and simulation represents an important opportunity for both the industry and the academia in the sense that, with a minimum consumption of resources, it allows explorations to be made on the emulated system as if it was the real system, supporting decision-making and projecting scenarios that in most cases are unexplored or inaccessible due to the high costs involved.

An agile iterative model, like the one proposed by Craig Larman, represents a good starting point for a research work around the modelling and simulation of software processes, so it is chosen in this work as the "ideal reference" model of the process to be simulated. Concerning the conceptual basis chosen for the modelling and simulation of this "real" system, system dynamics will be used because of the extensive available documentation on its use and exploitation and because of the added value that as a systems thinking approach provides.

With this in mind, here we propose to use a simulation model of an agile iterative software process as a tool to support the teaching and understanding of the software process dynamics, giving continuity to previous contributions [4–6] but emphasising on the usefulness of this approach in the process of predicting in advance the performance of a software process in different scenarios. This is explained in detail in the following sections.

## 2 | STATE OF THE ART

The work of García-García et al. [7] suggests that addressing problems from a process perspective has arisen in several fields of business process management (BPM), but in software engineering, the complexity is so high that it is necessary to go

beyond the automation of processes due to "*highly-complex tasks which could not be effectively automated with reduced costs*" [7, 8]. In the last decade, simulation modelling applied to the software process gained interest among software engineering professionals, who have increasingly worked on research using mainly agent-based models, system dynamics models, and less frequently discrete-events simulation.

A recent work of García-García et al. [7], presented a literature review on this topic and found 8070 articles published in this field by a systematic search in 4 digital libraries in the period 2013–2019, suggesting the existence of an interest that justifies research on this. Following, a brief mention of the most relevant works addressing the software process from a simulation perspective is given. Special attention has been paid to those works using agent-based modelling and system dynamics.

### 2.1 | Advances using agent-based simulation

In 2014, Nassal [9] presented a methodological approach for developing "*academical simulation games of software project management*", aiming to optimise the assignment of tasks among a determined set of developers "*by giving the right task to the right developer at the right time*". In this model, the agents of the model constitute project members as self-acting agents, while the artefacts to be used are modelled as resources available for the agents. The set of factors considered as affecting the software process in this work is limited.

In 2014, Honsel et al. [10] presented a proposal based on mining software repositories. The model focusses on system growth, bugs' lifetime and developer activity. It is aimed at evaluating the interplay of different possible future scenarios to improve the quality of software projects. In 2015, Honsel [11] presented another work about an agent-based simulation tool to predict the behaviour of the software project (regarding the temporal dimension). This simulation is based on the evaluation of ongoing processes and possible development trends at several points in time.

In 2016, Honsel et al. [12] developed another work on an agent-based simulation model that considered the commitment of developers and the effect that it has on the process' behaviour, defending the theory that the process' behaviour is mostly dependent on the developers' commitment. It is interesting that in this proposal they consider parameters like the effort consumption or the size of the project to be developed. Ahlbrecht et al. [13] presented a scalable agent-based simulation platform which allows the prediction of the behaviour of a software development project in terms of costs, resources and activities. Rubio et al. [14] presented a multi-agent system that implements an auction mechanism for simulating task allocation in open-source software (OSS) projects. The authors aimed to optimise the task allocation and reduce rework during the project execution.

In 2018, Mohammed Ali et al. [15] worked on the definition of an agent-based simulation model which allows the representation of real-world aspects to measure how the

software evolves in maintenance phases. Hurtado et al. [16] presented a work evaluating the application of modelling and simulation techniques to improve the management of projects which use the Extreme Programming (XP) methodology. They consider the levels of expertise, the size of the team, the expertise of each member, the salaries, the number of tasks, or the estimated duration as factors of importance.

Wysocki and Orlowski [17] proposed in 2019 a scrum-based methodology using a multi-agent simulation system to meet the requirements of stakeholders. They have also implemented multi-agent modelling to extract meta-data and evaluate the SCRUM and RUP software production processes and the appropriate methods and tools for project planning.

## 2.2 | Advances using system dynamics

In 2013, De Sousa et al. [18] presented a system dynamics (SD) model to analyse 5 techniques (ad hoc, checklist, perspective, scenario, N-fold) that can be applied to analyse the inspection phase in software projects. The model allows the estimation of the total number of defects from the early stages of software development as well as the effectiveness and cost of inspection activity. The parameters evaluated are related to the number of pages, the average number of defects per page, the average number of defects detected by inspectors, the number of pages per inspection, the number of inspection, the process' maturity level, the number of parallel teams or the number of inspectors per team.

In 2014, Matalonga et al. [19] presented an SD simulation model of a software factory production line. They aimed to analyse the changes in behaviour when selecting one among several alternatives, that is, they focussed on how useful SD is in helping to explore scenarios that are the result of changes in factors affecting the process.

In 2015, Saremi and Yang [20] proposed a simulation model to evaluate the performance of software developers in crowdsourcing software projects. This simulation considers more than 20 variables (registered worker, active worker, winner worker, company's reputation, tasks, tasks taken, tasks submitted, associated tasks, drop taken, task duration, etc.) which are associated with a software worker's behaviour in a crowdsourcing platform, the task uploading behaviour in the platform and the influence among software workers and uploaded tasks. Also in 2015, Hurtado et al. [21] presented an SD simulation model to help in decision-making in the scope of usability evaluation and quality on software user interfaces. This model improves the configuration of the software tester who must execute software testing processes in terms of usability evaluation.

In 2017, Alexandros et al. [22] presented a dynamic simulation model to compare agile methodologies, supporting the decision-making problem of deciding which of these approaches is more appropriate to be used in a collaborative software project with specific resources, planning, and costs.

In 2018, Orta et al. [23] proposed a decision-making framework to build simulation models aimed at improving decision-making within the Information Technology Infrastructure Library (ITIL) context. Their framework aims to systematically build simulation models and solve real-world organisation problems applying ITIL recommendations.

From the perspective of information systems, a very interesting contribution is given by Abdel-Hamid and Madnick [24] and Chin et al. [25]. [24] who in the late 80s and early 90s introduced system dynamics as a tool to understand project dynamics in information systems, considering many operative factors such as the effect of overtime spent by developers and also considering the importance of the intangibility of software, such as its invisibility. A pioneer in its field, this research evolved towards more formal contributions to software engineering processes such as [25], where a specific simulation model is presented, focussing on "*various software life-cycle development activities and management decision-making processes*" and keeping an operational view. This work is of supreme value to this research since it introduces system dynamics to the software process; however, due to both the evolution of the software field and the simulation approaches, the use of system dynamics in software engineering needs to be updated and considered from a high and strategic level for which this research is an input.

Although it is very interesting, no recent and relevant works were found to be focussing on using system dynamics to represent a specific software development process and considering as well an extended and consistent set of factors explicitly affecting the process. This is why the following proposal is relevant, since it will incorporate into the modelled and simulated process a wide set of parameters (or attributes) that in real life have an important effect on the process but which are not usually analysed as affecting factors, mainly due to the lack of a systemic view of the process or because they are considered to be out of the scope of the software management field. However, we think these factors are important and must be understood and considered.

## 3 | SIMULATION OF THE SOFTWARE PROCESS DYNAMICS

The simulation model developed by the authors (corresponding to the causal diagram of Figure A1, Figure A2 and the model views of Figure A3, Figure A4, and Figure A5 in Annex A) in order to perform the experiment that will be described in this work, represents the dynamics of a software development process and the factors that may affect such dynamics. For the development process the Craig Larman model is used, while for the representation of the factors affecting the process dynamics, the intermediate COCOMO II model's cost estimation drivers are used. Details about the model structure and the specific elements that it contains and how they can be used in software estimation simulation models can be found in [6], while the source [26] still remains useful for all purposes of

teaching and estimation, as may be seen even in recent years, such as in works related to the estimation of the effects of an audit review effort on project outcomes and project performance given by [27] as an example.

System dynamics (SD) was selected as the simulation paradigm for building the model. SD provides the whole conceptual framework for the construction of the model; thus, it invites the building of the simulation model from levels (or accumulations), flows and feedback cycles, which may be present and identified in the real system and is crucial to the proper functioning of the simulation model. The simulation model presented here is an evolution of the original partial model presented in a previous work [6], and the whole system dynamics model for simulating the software development process as well as its validation can be found at <https://promise.sel.inf.uc3m.es/images/files/SWProcessModel/SWProcessSDModel.zip>.

## 4 | DESIGN OF THE EXPERIMENT

In order to explore the effect of introducing system dynamics as a new paradigm for software engineers to learn and understand the dynamic behaviour of a software development process, an experiment was carried out. This experiment consisted of a 90 min study in which the 59 participants, junior software engineers (in the last course of their studies of

computer science), followed the steps given in Figure 1. The detailed description of the information that the participants in the experiment received can be found at <https://promise.sel.inf.uc3m.es/index.php/resources?view=article&id=37:system-dynamics-and-the-software-process&catid=13>.

**The first step** of the study consisted of a reading to understand the main features of the Craig Larman software development process, its phases, sub-phases, and the natural behaviour of the process execution in terms of time, workloads and the factors that may affect such behaviour. All participants already knew the Craig Larman process. A list of the COCOMO II model drivers (software, hardware, personnel and project) and their attributes was provided as well as a brief explanation of their effect on the software process.

**The second step** consisted of a form-filling task. This form contained four questions, each of them expressing the modification of two attributes that affect the software process and asking the respondent to select one among four possible affirmations about the effect that these modifications may have on a specific aspect of interest. Table 1 presents a classification of the levels of complexity that the questions may have had according to several factors: the number of attributes of the COCOMO II model that may be modified, the number of drivers (hardware, software, personnel or project) related to such attributes, and the effects observed through these questions. This classification will be useful for the discussion

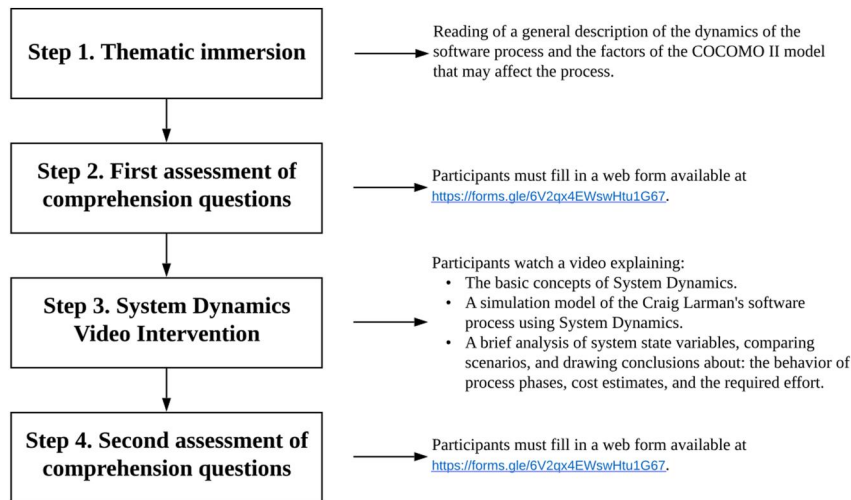


FIGURE 1 Design of the experiment

Level of complexity	N° of attributes modified	N° of drivers	Effect observed
L1	2	2	Total project execution time
L2	2	2	Displacement of graphs of specific phase workload
L3	2	2	Displacement of graphs of specific phase workload and workload width (duration of phase)
L4	2	2	Project cost estimation
L5	2	1	Effort consumption on a specific phase

TABLE 1 Factors affecting the complexity of the assessment questions



presented ahead in section 5. All participants were familiar with the estimation methods' drivers. For reasons of space limitation, the full form with the questions can be found in the first part of the form available at <https://forms.gle/6V2qx4EWswHtu1G67>; however, this first set of questions is given in Annex B.

**The third step** consisted of an intervention through the observation of a video. This 20-min video introduced system dynamics' basic concepts, structures and terminology. It was aimed at providing software engineers, who have never worked with system dynamics, with the wide strategic perspective that it involves for analysing situations and for problem-solving. For doing so, the video illustrated the use of system dynamics for understanding a system that was familiar to everybody; the dynamics of a population, with its related births and deaths flows. It also illustrated how environmental factors may be added to increase the real representation of the system. Finally, the video introduced a system dynamics simulation model representing the Craig Larman Software development process, emphasising on what the relevant stocks are (effort, analysis, design, implementation, and testing accomplishment) and how these may have inflows or outflows that contribute positively or negatively to such stocks' accumulations. The video also highlighted the power of simulating and using the graphs generated on the stocks and flow to analyse and understand the behaviour of the system in time and the prediction of the effect of the factors on such behaviours. This video is available at <https://youtu.be/jHmc16jX7ok>.

**The fourth step** consisted of a second form-filling task. This form contained eight questions—the same four questions that were given in step 2 and four additional questions asking for similar aspects, involving similar modifications to the factors of the software process that was explained; however, they were a bit more complex. The idea is to have enough information to discover whether the video observation has a positive effect on the participants' ability to guess the behaviour of the software process. For reasons of space limitation, the full form with the questions can be found within the second set of questions available through

the form at <https://forms.gle/6V2qx4EWswHtu1G67>; however, questions five to eight are given also in Annex C.

The previously explained experiment will allow us to explore the validity of the following hypothesis:

*“By introducing System Dynamics as a means for understanding the software development process, software engineers can better foresee the effect of variations on the drivers on the process dynamics”.*

## 5 | DISCUSSION ABOUT THE RESULTS OF THE EXPERIMENT

Following the experiment, first, a quantitative analysis of the results of the surveys that were carried out through the experiment previously described is presented and second, a qualitative analysis of the rationale the participants used to justify the estimations of system behaviour is presented.

### 5.1 | Quantitative analysis

The accuracy analysis of the respondents before and after the introduction of the system dynamics simulation model of the software process is presented next.

#### 5.1.1 | Accuracy before vs. Accuracy after (video observation)

Table 2 presents the accuracy values for the answers obtained both before and after the video intervention.

As it may be observed, the respondents hardly ever changed their minds regarding the first four questions in the second assessment after the first one.

##### (a) Question 1

In the case of Q1, the participants provided the correct answers in 100% of the cases both before and after the intervention video.

**TABLE 2** Accuracy before and after the video intervention

Question	Complexity	Before intervention	After intervention	Coincidence before-after
Q1	L1	100%	100%	100%
Q2	L1	96.61%	96.61%	96.61%
Q3	L2	96.61%	96.61%	100%
Q4	L1	11.86%	6.77%	89.83%
Q5	L3	--	79.66%	
Q6	L3	--	100%	
Q7	L4	--	15.25%	
Q8	L5	--	81.36%	

*(b) Question 2*

Regarding question 2, the absolute accuracy is the same before and after the video intervention; however, 2/59 of the respondents (3.39%) changed their minds from the first to the second survey, letting the researchers know that they were not totally sure of the appropriate answer. One of them was right and went wrong, while the other was wrong and went right.

*(c) Question 3*

In the case of question 3, the accuracy is the same (96.61%) before and after the video intervention, with no one changing his/her mind from one survey to the other, suggesting that it was a very simple question with no doubts emerging.

*(d) Question 4*

Regarding question 4, the results are very intriguing. The accuracy is very low both before (11.86%) and after (6.77%) the video intervention. Nonetheless, the most interesting effect is that around 11% (6/59) of the respondents changed their minds after the video intervention but not necessarily to switch from a wrong answer to a correct answer. So far, out of 6 respondents, three of them were correct before but were wrong in the second chance. However, as it will be explained later, the qualitative analysis revealed that this lack of accuracy may be related to the misunderstanding of one of the variables that are manipulated in this question (the PVOL).

### 5.1.2 | Effects observed on additional questions after the video intervention

The second survey carried out was aimed at collecting information regarding how increasing the complexity of the questions may affect the judgement of the respondents. The results are very interesting since they drive to a critical discussion on the use of system dynamics in this field, which is the main goal of the experiment.

*(a) Question 5*

Regarding question 5 (part of the post-video-observation assessment), 47/59 of the respondents (79.66%) were able to provide the correct answer. In this question, there were two affecting factors: the required software reliability extent (RELY) being improved and the required development schedule (SCED), remaining nominal.

Since the respondents were asked the same question about the execution of the test and its starting point and extension in time, the main effect would be given by the RELY attribute. The correct answer was, *“Testing starts occurring earlier (provided that the coding is faster) and takes shorter time”*. However, 12/49 respondents were still not able to guess it correctly, with equally distributed wrong answers among two alternatives: *“Testing starts occurring earlier (provided that the coding is faster) but takes longer time”* with 6/59 respondents, and *“Testing starts occurring later (provided that the coding is slower) but takes shorter time”* with 6/59 respondents, which

does not make much sense as the high reliability of the software facilitates the coding phase and decreases the probability of errors in the testing phase.

*(b) Question 6*

Concerning question 6, while having the same level of complexity as question 5, it showed much better performance on the estimation done by the respondents, since 100% of the participants were able to provide correct answers. The participants were asked to think about the effect of improving MODP (application of software engineering methods) and LEXP (programming language experience) in the implementation phase. All of the respondents could guess correctly that such a phase, under these conditions, is expected to start occurring as expected and takes a shorter time. In this case, the respondents were asked to think about how two positive changes, that is changes that are supposed to help the process execution, may be affecting a specific phase: the implementation. All of the respondents were able to estimate that by improving the “application of software engineering methods” (MODP), and the “programming language experience” (LEXP), the implementation phase is expected to start occurring as expected (not earlier or later) and take a shorter time.

*(c) Question 7*

There is an interesting result in the responses to question 7. Only 9 respondents out of 59 were able to provide the appropriate answer (15.25%). This is interesting since the only different aspect if compared to the previous questions, was that instead of asking people to predict the effect of changes in the attributes on the project execution, the question was about the effect on project cost estimation. It seems that for the respondents it was difficult to clearly identify the relationship that may exist between a project schedule compression and the costs that such compression involves, that is, for being able to compress the project execution time, there must be a monetary investment that directly increases the project costs. This indirect and not-so-obvious relationship between project compression and the costs that such compression may imply is hard to guess for junior professionals, while for senior professionals it is a matter of daily work and so it becomes clearly evident. Most of the participants focussed on the formal definition of the variables, which may have biased the sense-making work while deciding the appropriate answer.

*(d) Question 8*

For question 8, despite the complexity of the question, the accuracy of the responses is quite better. 81.6% of the respondents (48 out of 59) were able to estimate the effect on “effort consumption” that the modifications in two attributes had, which were quite familiar to them since these are directly observed in their daily activity: the capability of the analysts involved in the project (ACAP) and the capability of the software engineers (PCAP).

This question implied the task of identifying in which phase of the process both the work of the analyst (i.e. on the

analysis and design, mainly) and the software engineers are reflected (on the implementation and testing).

Although it looks very easy, the main difference in this question, if compared to the previous simpler questions, is that one of the attributes was changed to help the process improve (ACAP increases) while the other was changed to worsen the process (PCAP decreases), which may be the reason why 11 out of 59 respondents were still not able to guess the appropriate answer.

From a quantitative perspective, and considering only the accuracy of the responses, the results are not conclusive to strongly affirm that the system dynamics intervention video significantly helped the software engineers to better guess the effects of changes in the attributes of the software on the strategic aspects of importance: time execution, effort consumed and phases accomplishment. However, it is still a good possibility, and it is important to mention that in the second assessment the complexity of the questions was higher, and the accuracy of the respondents was higher than

expected. Actually, as said before, the only question of the second assessment in which the majority of the participants failed (Question 7) is justified by the fact that the participants did not have enough experience to see the relationship between project compression and the costs that it may imply.

## 5.2 | Qualitative analysis

The main difference between the first and the second assessment lies in the fact that the software engineers were able to provide a better justification for their responses in the second assessment's additional questions. Although the responses to these questions, in general, had a slightly lower accuracy, for those that provided a correct answer, they were in general more capable of explaining the dynamics of the process, making explicit reference to terms related to the software process dynamics. Table 3 and Table 4 show the percentages of

**TABLE 3** Statistics about the quality of the justifications given to the responses—Part 1

Question	Accuracy before intervention	N° of justified rationales	Quality of the rational		
			P = Poor	A = Average	G = Good
			<i>P</i>	<i>A</i>	<i>G</i>
Q1	100%	30/59	0/30 0%	27/30 90%	3/30 10%
Q2	96.61%	30/59	1/30 3.33%	1/30 3.33%	28/30 93.33%
Q3	96.61%	24/59	2/24 8.33%	0/24 0%	22/24 91.66%
Q4	11.86%	46/59	24/46 52.17%	16/46 34.78%	6/46 13.04%

**TABLE 4** Statistics about the quality of the justifications given to the responses—Part 2

Question	Accuracy after intervention	N° of justified rationales	Quality of the rational		
			P = Poor	A = Average	G = Good
			<i>P</i>	<i>A</i>	<i>G</i>
Q5	79.66%	34/59	2/34 5.88%	4/34 11.76%	28/34 82.35%
Q6	100%	24/59	0/34 0%	0/34 0%	24/24 100%
Q7	15.25%	43/59	16/43 53.48%	23/43 37.20%	4/43 9.30%
Q8	81.36%	33/59	5/33 15.15%	5/33 15.15%	23/33 69.69%



the respondents that were able to provide a rationale to their decisions and the grading that such responses had using the poor-irrelevant-good (P-I-G) scale.

The participants of the study were capable of providing a rationale justifying their selection in many cases; however, the rationale was not always good. There are several interesting and counterintuitive results.

The analysis of questions 1–4 has been omitted in the second assessment due to the valence effect resulting from the fact that they already knew these questions from assessment 1.

### 5.2.1 | First assessment

In the first assessment, although for questions 1, 2 and 3 the responses were correct in most cases (100%, 96% and 96%, respectively), only a portion of these provided a rationale for their responses (30/59 for Q1, 30/59 for Q2, 24/59 for Q3 and 46/59 for Q4). What follows is a brief analysis of these responses.

- For the first question, most of the justifications provided (90%) were “average” and only 10% were “good”. This allows the authors to think that there is an effect of the order of this question on the respondents, since they were not accurate enough even though the question was very simple (L1 level of complexity). But exploring this effect was not the goal of this study.
- For questions 3 and 4, the accuracy was high (96.61% in both cases). However, only a portion of the participants (30/59 for Q2 and 24/59 for Q3) provided a justification. The quality of the justifications is high for these questions, showing a good quality for the biggest portions (93.33% and 91.66%, respectively).
- It is interesting that there is high participation in providing a rationale for question 4. However, only 11.86% of the total participants were accurate in this question. So, why is there a high participation in this? The response is that most of the participants made an effort to justify their response by providing extra information, driving the authors to realise that they were confused about the meaning and sense of one of the COCOMO drivers that was being modified: the PVOL. The text of the justifications shows that many people confused the semantic meaning of volatility in the development platform. While the PVOL represents the probability of causing a system failure, many respondents interpreted the opposite and thought that it was the probability of not failing.

### 5.2.2 | Second assessment

In the second assessment, the general accuracy is lower than in the first assessment. This is something that the authors were expecting, since the complexity of the questions is higher as well. The first four questions of this assessment were the same as in the first assessment, so they will not be analysed next.

- For question 5, the accuracy of the respondents was not bad (79.66%); however, it could have been better. Out of the 59 respondents, 34 provided a justification. An important high number (82.35%) of respondents out of these 34 were able to provide a good quality, correct justification, while 11.76% of the responses were of average quality and 5.88% of poor quality.
- Regarding question 6, although 100% of participants responded accurately, only 24 out of 59 participants provided a justification. But for sure, all of them (100%) were of high quality. In this case, the level of complexity in terms of the COCOMO drivers modified was not the simplest, and the asked effect was double (starting point and duration of the project execution). The accuracy of the responses may be attributed to the fact that the parameters that were modified are familiar to any software professional with minimum experience: application of software engineering methods (MODP) and programming language experience (LEXP); so, it did not require additional effort to comprehend the semantics of these attributes.
- The responses to Question 7 were not accurate at all. In this question, the respondents were asked to think about the effect of decreasing the “Volatility of the virtual machine environment (PVOL)” and consider a “Required development schedule (SCED)” on the estimation of costs. Only 15.25% of respondents selected the correct option—that the costs increase. Most of the respondents (53.48%) mentioned in their justification that the costs would be lower than in the nominal case. This result may be explained by mentioning that, as in the first assessment, the semantic of the attribute PVOL is not clear for the software engineers, since they understand that a lower PVOL involves a high probability of system failure (an inexpensive option). So the cost would be lower. A decrease in the PVOL combined with a schedule constraint is related to a higher cost.
- Concerning question 8, 33 out of 59 respondents justified their response. Out of these 33 participants, 23 (69.69%) were able to provide a good rationale using system behaviour terminology.

From a qualitative perspective, considering that the complexity of the questions in the second assessment was higher than the complexity of the questions in the first assessment, the results suggest that there is a positive effect of the system dynamics intervention video on the respondents’ responses. In general, the quality of justifications (rationale) for more complex questions was nurtured with quality and terminology related to strategic analysis of the software processes, making the participants create connections of causalities among the phases of the software process.

## 6 | CONCLUSIONS

In this work, system dynamics was used as an approach for the construction of the software process model, based on the fact that the software process, far from being methodical and

procedural turns out to be complex, variant, and in many occasions counterintuitive, with many emerging and unpredictable phenomena.

For the construction of the model that was illustrated during the experiment, and according to the methodology taken as reference [28], first, a deep immersion in the system to be modelled was made, which translated into a deep and systemic understanding of the software engineering process of the Larman method, its operation, phases, characteristics, and properties, constituting a *first level of abstraction of the process*. Part of this immersion was the exploration of approaches to describe the parameters that affect the process, for which it was very useful to make use of the COCOMO cost model which has defined a series of parameters common to the processes that have an impact on development and related resource consumption.

In the experiment and the related model presented here, these parameters were used as a way to increase the representativeness of the model, which allowed the incorporation of the effects of the process parameters on the operation of the process as a whole, and thus the estimation of possible effects and expected behaviour from certain parameter configurations. This property represents a significant contribution for those who want to study the software process with minimal costs and in a practical, manageable manner that is usable by multiple users and has significant potential to support decision-making in software process management teams.

The good immersion in the operation of the software process and the well-founded identification of the parameters that affect the process served as the basis for the construction of the representative causal diagrams about the software process, initially of the whole process and later, as a sort of zoom within the general system of the construction phase of the software engineering process. The main contribution in this aspect came at the time of modelling this system (the construction sub-process) using the system dynamics approach. In this modelling, the identification of variables, levels, and flows that interact in the system represented a *second level of abstraction of the system of great importance and utility*, since the relationships between the elements of the system that are not always evident at first sight or with superficial exploration are identified. Subsequently, all the relationships, causalities, and feedback loops identified were implemented in the design, configuration, validation and manipulation of the model itself, achieving a fairly broad level of system representativeness and forming one of the main strengths of this work, being a representative model of the software process which considers a large number of elements and parameters of different types that inevitably influence the system and should not be ignored.

The experiment carried out using the model to illustrate the software process and the dynamics related to such a process to junior software engineers was very interesting. While trying to discover how useful the software process system dynamics model may be to help these professionals to better understand the process from a strategic view, the experiment allowed the authors to clarify the usefulness of system dynamics models in this field.

The results suggest that there are slightly, but not significantly better, quantitative results of their task of predicting the performance of the system when factors affecting the system (the software process) are manipulated.

However, regarding the qualitative analysis of these professionals' predictions, it seems that they gained vocabulary and quality of analysis at the moment of explaining their comprehension of the system's (the software process) behaviour.

In summary, from a quantitative perspective, the participants were not significantly better than before knowing system dynamics' basic concepts, but from a qualitative perspective the study suggests that the participants improved their ability to provide arguments about their guessing tasks and the choices made, even using terminology that may be used at a strategic level at the moment of discussing the behaviour of the software process and the attributes that more critically affect the whole process.

This experience caused arguments that emerged to support the claim that there is a need for a structured methodological approach to provide a context for studying problems similar to the one faced here: simulating software engineering processes and exploring the effect of such simulations on professional and academic environments. Facing these challenges would propitiate multiple benefits both for the sciences in general and for the fields of science involved in understanding organisational phenomena, including those of the software process. That is why it is proposed to give continuity to the present work in four ways:

- First, evolving the model developed here by experimenting with equations and function tables that may better represent the expected behaviours of the software process and its related sub-processes.
- Second, complementing the model presented here with models developed following other simulation approaches. This would allow contrasting approaches and defining differentiation criteria according to interests and use cases.
- Third, simulating software processes which are different from Larman's model, both more methodical (and therefore sub-representative of the process) and more organic and agile (therefore more adjusted) models, so that they are able to contrast and make significant contributions to the body of knowledge on software process modelling and simulation.
- Fourth, designing an experiment with more controlled variables, with a control group, and guaranteeing the absence of a valence effect among the different phases of the experiment.

Finally, it should be mentioned that this research contributes to the field of software engineering, through the provision of a tool and the simulation model developed, to predict and envision the behaviour of a software process without spending costly consulting tasks or testing on the road while the software is developed. The costs of a project are critical, and being able to simulate and know the effect of cost drivers can support early decision-making and enhance

the strategic discussion on a software process, which may also be useful in academic and educational contexts.

## ACKNOWLEDGEMENT

This work has been supported by the Madrid Government (Comunidad de Madrid-Spain) under the Multiannual Agreement with UC3M in the line of Excellence of University Professors (EPUC3M17) and in the context of the V PRICIT (Regional Programme of Research and Technological Innovation).

## ORCID

German-Lenin Dugarte-Peña  <https://orcid.org/0000-0001-9760-7084>

## REFERENCES

- Vicente, R.: Modelamiento semántico con Dinámica de Sistemas en el proceso de desarrollo de software. *Iber. J. Inf. Syst. Technol.* 10, 19–33 (2012). <https://doi.org/10.4304/risti.10.19-34>
- Robertson, S.: Learning from other disciplines [requirements engineering]. *IEEE Software*. 22(3) (2005). <https://doi.org/10.1109/MS.2005.68>
- Larman, C.: *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, 3rd edn. Prentice Hall, New Jersey (2005). <https://www.oreilly.com/library/view/applying-uml-and/0131489062/?ar>
- Dugarte-Peña, G.-L.: Software engineering under the prism of System Dynamics. XXIII Jornadas Internacionales de Ingeniería de Sistemas. Universidad Católica de Santa María, Perú (2016)
- Dugarte-Peña, G.L.: Modelado y Simulación de un Proceso de Desarrollo de Software dirigido por el Método de Craig Larman: Una aplicación de la dinámica de sistemas. (Modelling and Simulation of a Craig Larman Methods' Software Development Process: a System Dynamics Approach). Universidad Carlos III de Madrid, Madrid (2015)
- Dugarte-Peña, G.-L., et al.: Simulation of the software development process: an approximation using System Dynamics and the Larman Method/Simulación del proceso de desarrollo de software: una aproximación con Dinámica de Sistemas y el Método de Larman. *Rev. Innovación y Softw.* 1(1), 39–57 (2020). <https://revistas.ulasalle.edu.pe/innosoft/article/view/11>
- García-García, J.A., et al.: Software process simulation modelling: systematic literature review. *Comput. Stand. Interfaces.* 70, 103425 (2020)
- Papazoglou, M., Ribbers, P.: *E-business: Organization and Technical Foundations*. Wiley (2006). <https://www.wiley.com/en-gb/exportProduct/pdf/9780470064467>
- Nassal, A.: A general framework for software project management simulation games. In: *Iberian Conference on Information Systems Technologies (CIST)*, pp. 1–5. IEEE, Barcelona (2014). <https://doi.org/10.1109/CISTI.2014.6877074>
- Honsel, V., Honsel, D., Grabowski, J.: Software process simulation based on mining software repositories. In: *IEEE International Conference Data Mining Workshop (ICDMW)*, pp. 828–831. IEEE, Shenzhen (2015). <https://doi.org/10.1109/ICDMW.2014.35>
- Honsel, V.: Statistical learning and software mining for agent based simulation of software evolution. In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2, pp. 863–866. IEEE, Florence (2015). <https://doi.org/10.1109/ICSE.2015.279>
- Honsel, D., et al.: Monitoring software quality by means of simulation methods. In: *International Symposium on Empirical Software Engineering and Measurement*. ACM, Ciudad Real (2016). <https://doi.org/10.1145/2961111.2962617>
- Ahlbrecht, T., et al.: Agent-based simulation for software development processes. In: *Criado Pacheco N., et al. (eds.), Agent-based simulation for software development processes*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9571, pp. 332–340. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-59294-7\\_28](https://doi.org/10.1007/978-3-319-59294-7_28)
- Rúbio, T.R.P.M., Lopes Cardoso, H., da Costa Oliveira, E.: MAESTROS: multi-agent simulation of rework in open source software. In: Novais, P., et al. (eds.) *Intelligent Distributed Computing: Proceedings of the 9th International Symposium on Intelligent distributed Computing – IDC'2015*, October 2015, vol. 616, pp. 403–413. Springer, Guimarães (2016)
- Mohammed Ali, S., et al.: Developing an agent-based simulation model of software evolution. *Inf. Softw. Technol.* 96, 126–140 (2018) <https://doi.org/10.1016/j.infsof.2017.11.013>
- Hurtado, N., et al.: Applying agent-based simulation to the improvement of agile software management. In: Mas, A., et al. (eds.) *Software Process Improvement and Capability Determination*, pp. 173–186. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-67383-7\\_13](https://doi.org/10.1007/978-3-319-67383-7_13)
- Wysocki, W., Orłowski, C.: A multi-agent model for planning hybrid software processes. *Procedia Comput. Sci.* 159, 688–1697 (2019). <https://doi.org/10.1016/j.procs.2019.09.339>
- De SousaCoelho, J.J., Braga, J.L., Ambrósio, B.G.: System dynamics model for simulation of the software inspection process. *ACM SIGSOFT Softw. Eng. Notes*. 38(5), pp. 1. (2013). <https://doi.org/10.1145/2507288.2507306>
- Matalonga, S., Solari, M., San Feliu, T.: An empirically validated simulation for understanding the relationship between process conformance and technology skills. *Softw. Qual. J.* 22(4), pp. 593–609. (2014). <https://doi.org/10.1007/s11219-013-9214-2>
- Saremi, R.L., Yang, Y.: Dynamic simulation of software workers and task completion. In: *Proceedings of the 2nd International Workshop on CrowdSourcing in Software Engineering*, pp. 17–23. IEEE, Florence (2015). <https://doi.org/10.1109/CSI-SE.2015.11>
- Hurtado, N., et al.: Using simulation to aid decision making in managing the usability evaluation process. *Inf. Softw. Technol.* 57(1), 509–526 (2015). <https://doi.org/10.1016/j.infsof.2014.06.001>
- Alexandros, N.K., et al.: Comparing scrum and XP agile methodologies using dynamic simulation modelling. In: Kavoura, A., Sakas, D.P., Tomaras, P. (eds.), *Strategic Innovative Marketing*, pp. 153–158. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56288-9\\_52](https://doi.org/10.1007/978-3-319-56288-9_52)
- Orta, E., et al.: Decision-making in IT service management: a simulation based approach. *Decis Support Syst.* 66, 36–51 (2014). <https://doi.org/10.1016/j.dss.2014.06.002>
- Abdel-Hamid, T., Madnick, S.E.: *Software Project Dynamics: An Integrated Approach*. Prentice-Hall, Inc. New Jersey (1991)
- Lin, C.Y., Abdel-Hamid, T., Sherif, J.S.: Software-engineering process simulation model (SEPS). *J. Syst. Softw.* 38(3), 263–277 (1997). [https://doi.org/10.1016/S0164-1212\(96\)00156-2](https://doi.org/10.1016/S0164-1212(96)00156-2)
- Boehm, B.: *COCOMO II Model Definition Manual*, vol. 87(5 Suppl), p. i (2012). <https://doi.org/10.4269/ajtmh.2012.875suppack>
- Agrawal, M., Chari, K.: Impacts of process audit review and control efforts on software project outcomes. *IET Softw.* 14(3), 293–299 (2020). <https://doi.org/10.1049/iet-sen.2019.0185>
- Sterman, J.: *Business Dynamics: Systems Thinking and Modelling for a Complex World*. McGraw-Hill, USA (2000)

**How to cite this article:** Dugarte-Peña, G.-L., et al.: Using system dynamics to teach about dependencies, correlation and systemic thinking on the software process workflows. *IET Soft.* 15(6), 351–364 (2021). <https://doi.org/10.1049/sfw2.12031>

## APPENDIX

## Annex A

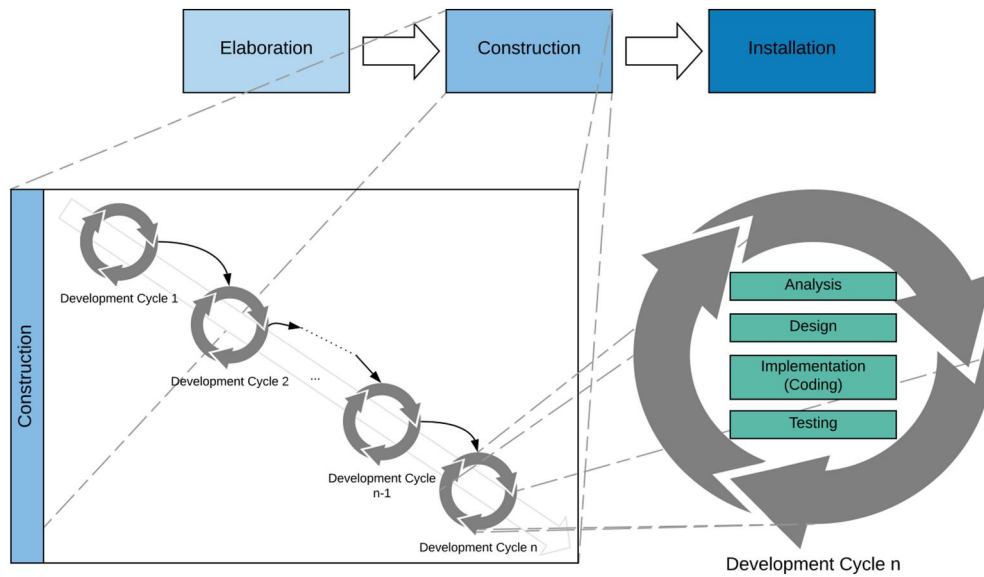


FIGURE A1 Craig Larman's software development process [6]

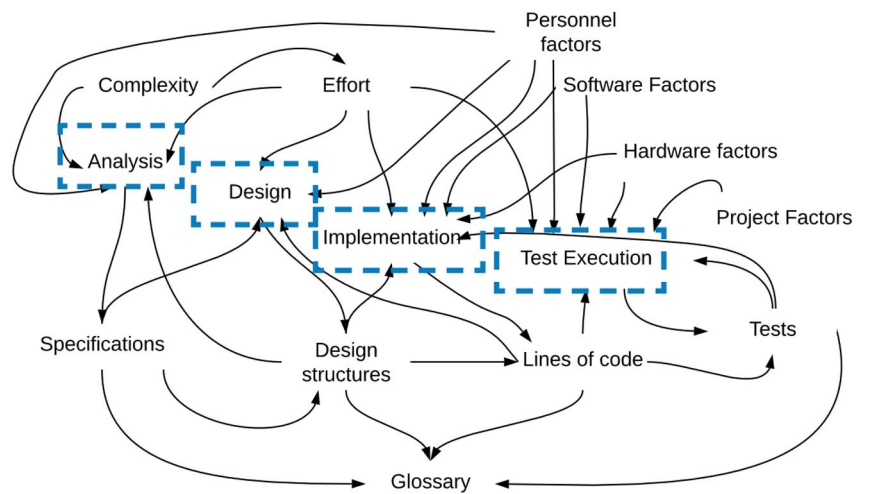


FIGURE A2 Causal Diagram - The Larman's Software Development Process [6]

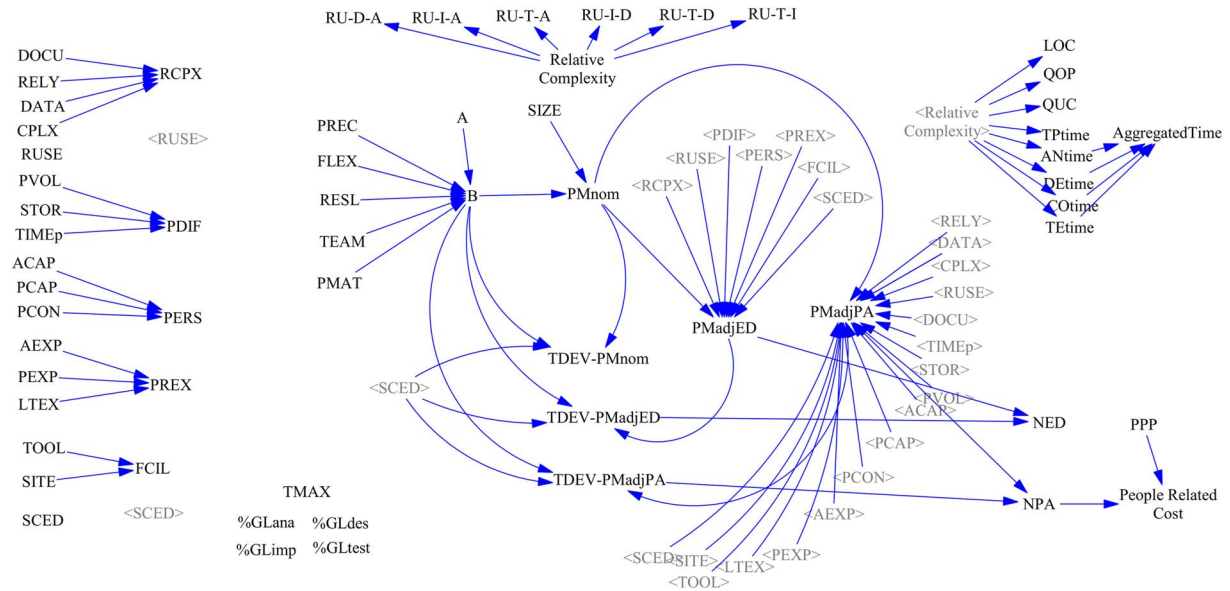


FIGURE A3 Model view 1: Software process attributes and cost drivers [6]

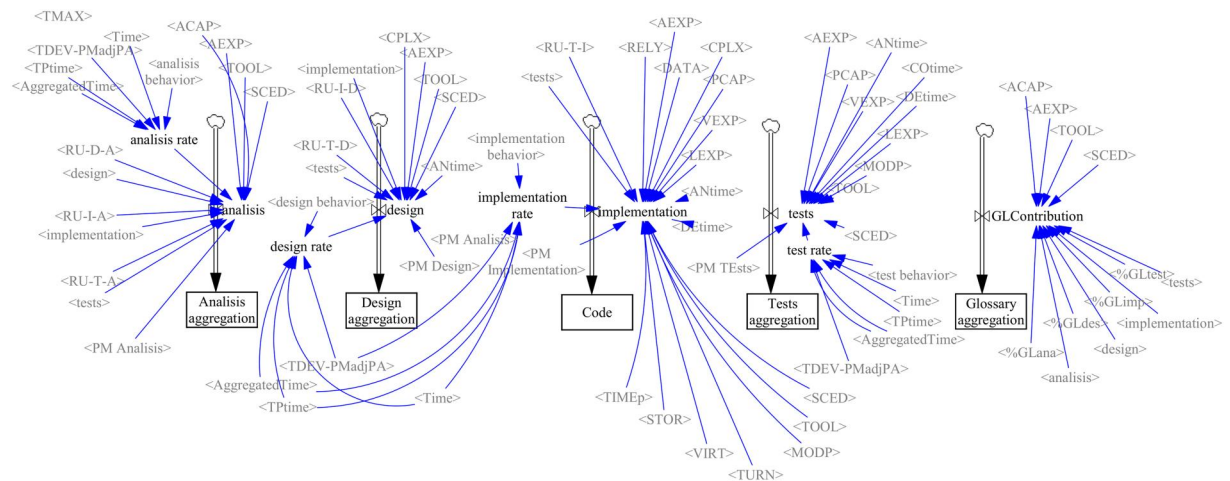


FIGURE A4 Model view 2: Accomplishment of phases [6]



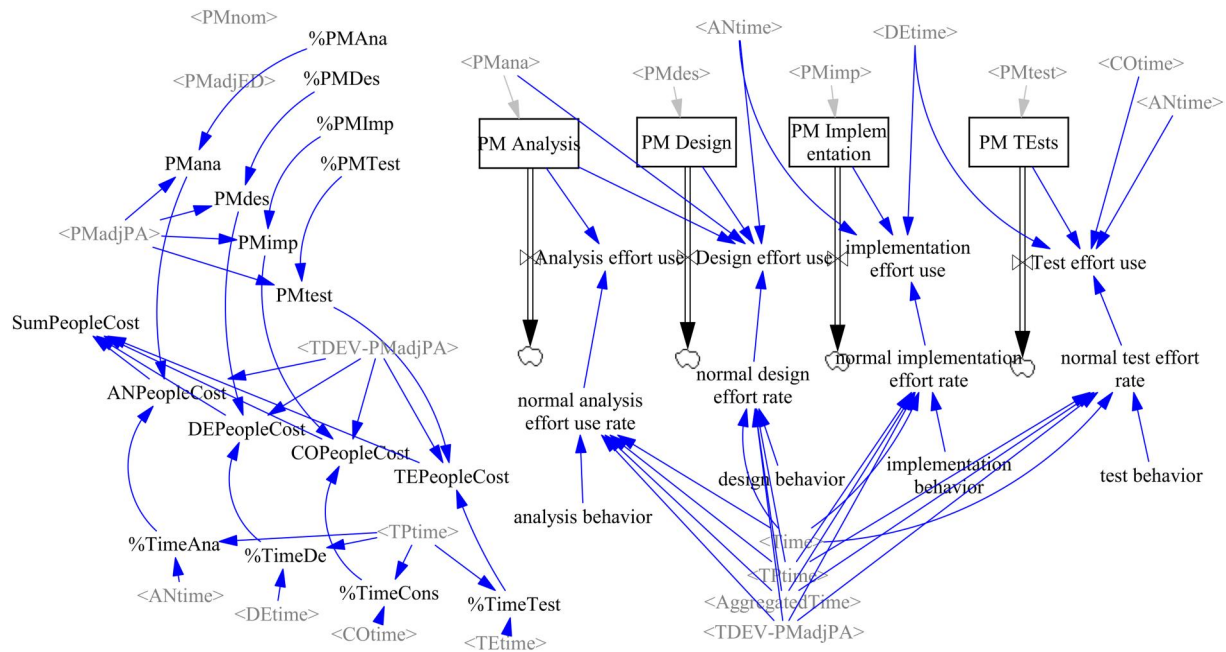


FIGURE A5 Model view 3: Effort consumption during the software process [6]

## Annex B

This annex presents the questions Q1–Q4 of the first assessment described in section 4.

After every question, there was an additional space for the respondents to provide an explanation or justification for their responses, with the specific text: “*You can use this space just in case you want to justify your reasoning around the answer you have provided for the above question. Otherwise leave it empty*”.

1. If the “Capability of project analysts” (ACAP) worsens and the “Complexity of system modules (CPLX)”, that is the complexity of the software platform used in the process of development (CPLX) increases, how do you think that a cycle of the project completion will be affected?
  - The project execution takes LESS time
  - The project execution takes MORE time
  - The project execution takes THE SAME time
  - I do not know
2. If the “experience of the analyst in domain related to the project”, that is how familiar the domain is to them, (AEXP) worsens, and the “Use of software tools” (TOOLS) decreases, how do you think that the project completion will be affected? \*
  - The project execution takes LESS time
  - The project execution takes MORE time
  - The project execution takes THE SAME time
  - I do not know
3. If the capabilities of the programmers (PCAP) improves, and the capacity of processing of the virtual machines used during the programing, known as “Memory constraints”

(STOR) improves, how do you think that the effort consumption will be affected? \*

- The effort on “Implementation” is shorter than expected.
  - The effort on “Implementation” is as expected.
  - The effort on “Implementation” is longer than expected.
  - do not know
4. If there is a worsening in the “Volatility of development platform” (PVOL), that is probability of causing a system failure during the coding sub-phase; and the project must adjust to an imposed “Development schedule compression” (SCED), that is there is pressure and the tasks to be developed are forced to fit in a shorter time; then the project execution is expected to: \*
    - Expand and thus take longer time
    - Compress and be completed in shorter time
    - Indifferent, the effect is not clear
    - I do not know

## Annex C

This annex presents the questions Q5–Q8 of the second assessment described in section 4.

After every question, there was an additional space for the respondents to provide an explanation or justification for their responses, with the specific text: “*You can use this space just in case you want to justify your reasoning around the answer you have provided for the above question. Otherwise leave it empty*”.

5. If there is an improvement on the “Reliability of the software development platform” (RELY); and there is no



special Development time constraints (SCED), how is the accomplishment of the Testing expected to vary? \*

- To start occurring earlier (provided that the coding is faster) but taking longer time
  - To start occurring earlier (provided that the coding is faster) and taking shorter time
  - To start occurring later (provided that the coding is slower) but taking shorter time
  - To start occurring later (provided that the coding is slower) but taking longer time
6. If there is an improvement on the “Up-to-date programing techniques” (MODP), that is the management team is up-to-date and applies modern and agile techniques; and the Experience in the language (LEXP) improves, the implementation phase is expected to: \*
    - Start occurring as expected but taking longer time
    - Start occurring as expected and taking shorter time
    - Start occurring later but taking shorter time
  7. If there is a decrease in the “Volatility of development platform” (PVOL), that is probability of causing a system failure during the coding sub-phase; and there is a "Development schedule compression" (SCED) of tasks, that is there is pressure and the tasks to be developed are forced to fit in a shorter time; then the project cost estimation is expected to: \*
    - Start occurring later but taking longer time
    - Be higher
    - Be lower
    - Be around the same
    - I am not sure/No idea
  8. If the “Quality of the programmers” (PCAP) worsens considerably; and the Analysis capacity (ACAP) improves, then the effort consumption is expected to: \*
    - Concentrate on Analysis
    - Concentrate on Design
    - Concentrate on Implementation
    - Concentrate on Testing