We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter 6

Technology Assessment in Software Development Projects Using a System Dynamics Approach: A Case of Application Frameworks

José Ignacio Muñoz Hernández,
José Ramón Otegui Olaso and
Alejandro Gutiérrez López

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/54498

1. Introduction

Technology is in constant change, new knowledge creates new technological opportunities; and there are always possibilities for improvement [1]. This constant innovation creates a plethora of technological alternatives from where an organization can choose and implement. In addition to this large array of alternatives, technologies are becoming more complex and evolving faster, making selection more difficult.

Technologies can create competitive advantages if they are selected correctly, this selection can be considered one of the most challenging decision making areas managers face [2]. The selections must match the organization previous technologies and systems, as well as other aspects of the organization such as human factors, culture, strategy and objectives. Organizations should be prepared for the broad spectrum of impacts when introducing new technologies, for instance the multiple costs associated and the social implications.

To make a correct selection, managers must carefully analyze the different technological alternatives. The objective of this assessment is to evaluate the consequences of introducing a technology. If the assessment is done before the technology adoption and investment, it reduces the risk of ineffective investment decisions [3]. When an organization integrates new technologies without being aware and prepared for its impact, it can lead to serious problems.

INTECH
open science | open minds

© 2013 Muñoz Hernández et al.; licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In the context of technology management, technology assessment (TA) can be defined as "a systematic attempt to foresee the consequences of introducing a particular technology in all spheres it is likely to interact with" [1], which can include the inside of an organization, project or process as well as the society. Because the technology has been not implemented yet, or may not even exist, the consequences have not occurred at the time of the assessment. In the technology assessment field, forecasting does not provide an assertion of the future, but a visualization of what it might be and the opportunities it may bring.

There are three basic forecasting methods: extrapolation, expert opinion and modeling. The extrapolation method involves obtaining historical data, fit a curve to them and extend the values into the future. When an expert opinion method is employed, experts in certain field try to reach a consensus on the likely future. Modeling consists in using mathematical formulas to represent the relationships that exist between variables in the real world.

One of the modeling approaches used in technology assessment is in a system dynamics. System dynamics is a "perspective and set of conceptual tools that enable us to understand the structure and dynamics of complex systems" [4].

In the software industry, developments are made in complex and dynamic systems involving several interrelated elements such as people, processes, methods, products, technologies, tools and techniques [5]. These interrelations create a feedback system where a change or improvement in one area creates effects in others parts of the system directly or indirectly. Additionally, system complexity might stem from its risks and uncertainties, dynamic behavior or changes over time [6].

Simulation techniques like system dynamics can be used to understand these interrelations and to analyze in advance the impacts of changes or improvements such as new technology adoption. In this chapter we describe the use of system dynamics models and simulations to assess technologies in the context of software engineering.

This chapter is organized as follows; in the first section we review the field of technology assessment and explain an intermediate level assessment using system dynamics. In the next section, the field of software process simulation modeling and a general overview of system dynamics are presented. The subsequent sections are composed of studies applied to technology adoption and a study regarding a reuse technology. At the end of the chapter conclusions and future research of this study can be found.

2. Technology assessment

2.1. Background

The field of technology assessment started in the 1950s with studies that attempted to forecast technological trends to help government agencies and large corporations in their technological investments; later, in the public decision domain of the United States, technology assessment started to focus on studying all the effects of technologies still to come [7].

http://dx.doi.org/10.5772/54498

There have been different approaches in time and in the public and private sector regarding technology assessment. However, there is a common concern about the current and future developments of technology and to improve the alignment between technological and societal developments, which includes the activities of corporations.

2.2. Application from a technology manager's perspective

The use of technologies is one of the main components that determine the success of an organization. From the technology manager's perspective, technology can be defined as "the ways and means by which humans produce purposeful material artefacts and effects" [1], this definition includes the material elements, such as hardware and means, but also the soft elements, such as: knowledge, ways and methods.

To take the best advantage from technology, a firm needs to analyze the technological alternatives and their consequences thoroughly. These consequences must considered not only in the short term but as far as possible and contemplate its implications to the full context, for the firm and the environment in which it operates. Technology assessment can provide the framework to create this type of analysis to evaluate the firm's products, processes and supporting technologies current and future positions.

The assessment of new technologies can be considered from two perspectives: the introduction of a new technology to improve a production process or the launch of a technology into the market. The former can refer to a process innovation, such as the development of new production equipment, new production processes and the introduction of new technologies in a process. The second, known as product innovation, refers to the development of new products for the consumer.

According to [8], technology assessment can provide managers information to:

Support decision making for:	Help the firm:
R&D direction	Corporate strategic technology planning
New technology adoption	Perform value chain analysis
Incremental improvement in existing technologies	Identify market advantage
Level of technology friendliness	Avoid being preempted in the marketplace
'Make -or-buy' decisions	Stay on the cutting edge
Optimal expenditure of capital equipment funds	Define the cutting edge
Market diversification	Maximize the use of information
	Minimize product makespan
	Achieve a competitive advantage (in cost, quality, or time to market)

Table 1. Examples of technology assessment application areas in the organization. [8]

A basic methodology for technology assessments can be outlined in the five steps described in Figure 1:

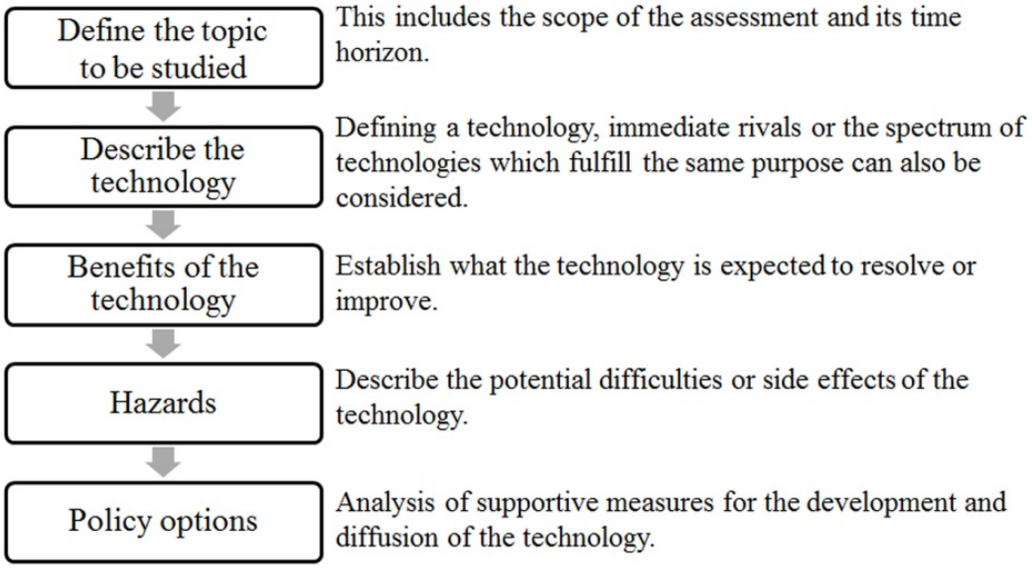


Figure 1. Diagram of the five basic steps in a technology assessment methodology

A crucial element in a technology assessment is the gathering of information relevant to the issue; it involves discussions and dialogue between multiple actors in its development. The information can reside in a variety of sources, including opinions, attitudes, fears, interests.

The possibility of using simulation to assess technologies has been emphasized in different works, in [9], it is asserted that performing simulations in organizations can help evaluate technology strategies; in [10], the same approach of using process simulation modeling to evaluate issues of technology and tool adoption is used.

One of the techniques that can be used in a technology assessment is system dynamics. It provides tools which can be used to include the multiple perspectives involved in the decision and analyze them. Using this technique, it is possible to adopt a holistic view of the company to understand it as a system, considering its products, processes and supporting technologies and enable the development of an intermediate level assessment of a technology [11].

2.3. Intermediate level assessment

Dynamic simulation models and maps of the anticipated domain of application of a technology make it possible to evaluate its impact at different levels of the organization, for example strategic and project levels. These simulations will assist decision makers and it can reduce their decision risks.

An intermediate level assessment of a technology can be created using a system dynamics model; it can capture the domain-centered orientation of low level approach but provide a balanced assessment of impact across the whole domain of application. This assessment will be located between a specific or low-level of a detailed analysis of the technology characteristics and functionalities, and a high level view provided by an economic analysis. In this

http://dx.doi.org/10.5772/54498

intermediate level it is necessary to create a model based on the technology and the domain where it will be used.

In [11], Wolstenholme suggests a three step methodology to create this intermediate assessment:

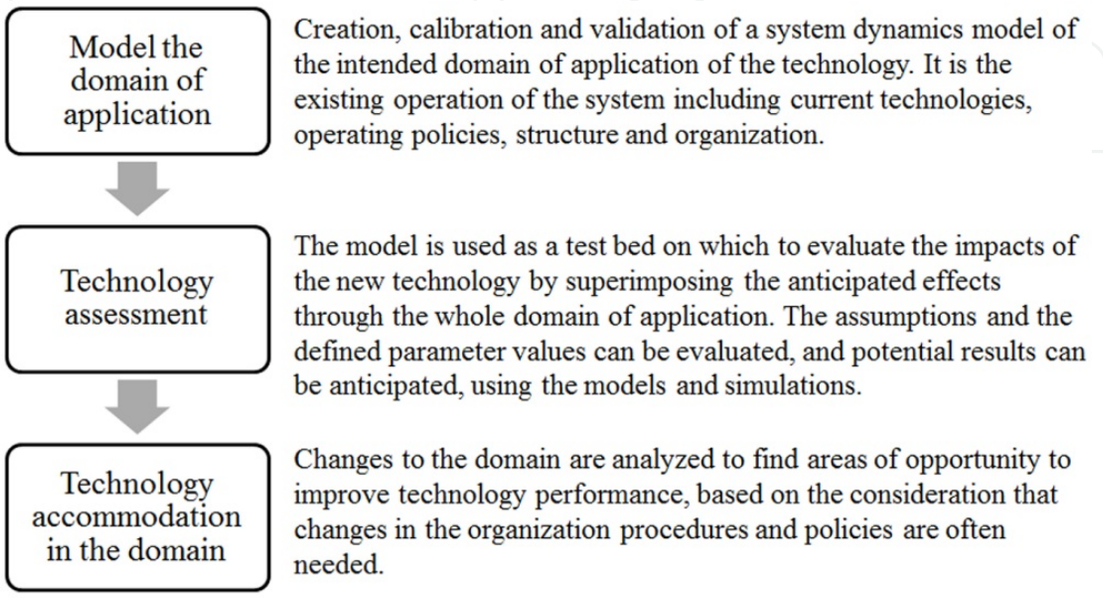


Figure 2. Diagram of the three step methodology to create an intermediate assessment.

Some adjustments that might be projected in the simulation are: changes in the organizational boundary, responsibilities of activities, elimination of delay, increasing or reducing capacities, information paths and policies, or reengineering processes.

These stages of the methodology proposed by Wolstenholme are related to the basic technology assessment methodology described earlier in the chapter. The definitions of the technologies, areas of application, expected benefits and difficulties, as well as the current and future policies, can serve as input in the creation of the models. At the same time, the knowledge acquired with the model simulations feedback the technology assessment.

3. Modeling and simulation in software

3.1. Software process simulation modeling

From the perspective of software research, the area focused on simulation and modeling of software processes and projects is Software Process Simulation Modeling (SPSM). SPSM can be helpful in the definition of why to use simulations, what to model and simulate, which simulation method to use, how to perform this activities and which parameters to use; it provides recommendations when using and implementing this type of analysis in an organization, as well as describe potential issues.

When performing a technology assessment on software engineering technologies, SPSM provides relevant information to execute the assessment, because it focuses purely on issues of software development and maintenance.

Concerning the why, Kellner et al. [6] identified and grouped the purposes for using simulations into six categories: strategic management, planning, control and operational management, understanding, training and learning, process improvement and technology adoption.

Process improvement and technology adoption can be grouped together in the same category because both support decisions for improvements (such as the priorization of various proposals) through forecasting of potential impacts; the difference is that the first explores the improvement of processes, and the second deals with the introduction of new technologies. These purposes can also be related to planning in the sense that they are concerned with the future, but the main difference to planning is that the objective is not to make a plan but to analyze an improvement effort.

Within the field of software process simulations there are various approaches to simulate software processes and projects. In [12], ten simulation paradigms were identified; however, system dynamics is the most widely used technique in SPSM.

Using system dynamics modeling and simulations, it is possible to [9]:

- Research about a great variety of aspects of a software process at a macro or micro levels.
- Evaluate different alternatives and make the selection based on their implications.
- Compare life cycle processes, defect detection techniques, testing strategies and different methodologies and tools.
- Represent systems and processes as they are currently implemented, or as-is, but also as planned or expected in the future, or to-be.

Two of the cornerstones of SPSM with system dynamics are the works of [9] and [13]. The difference between these two works is where each establishes the boundary of their view; [9] integrates processes outside the project boundary, [13] does not. The work in [9], as most of the work in the area, considers a process perspective. A software process can be defined as "a set of activities, methods, practices and transformations that people use to develop and maintain software and the associated products (e.g., project plans, design documents, code, test cases, and user manuals)" [14].

In order to determine what will be simulated, it is necessary to have the basic purpose for the simulation model. A simulation model is a computerized model used to represent a dynamic system. When a simulation model is being developed, it is necessary to identify the main elements of the process under study, interrelations and behaviors to abstract the process. Along with the specific issues to be analyzed, the scope of the model, the input parameters and the resulting variables and an abstraction of the process will also have to be defined. A diagram of these interrelations between the aforementioned elements can be seen in Figure 3.

http://dx.doi.org/10.5772/54498

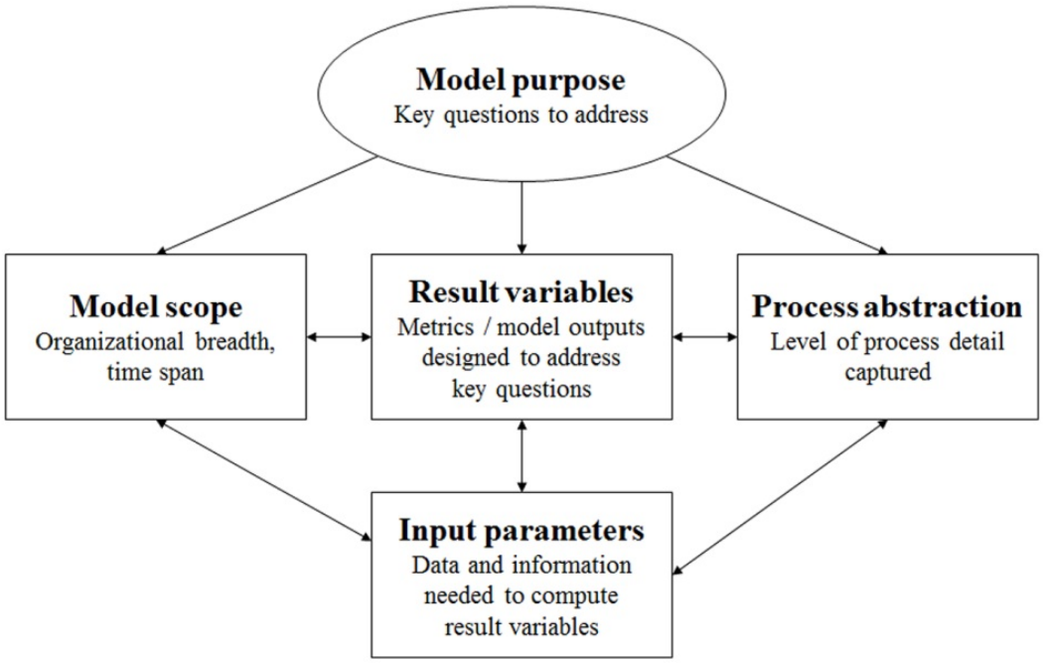


Figure 3. Diagram showing the interrelationships between model elements [15].

Usually, the scope of the simulation is among the following: a portion of the life cycle, a development project, multiple concurrent projects, long term product evolution and long term organization. It is necessary to define the timespan, which can be under twelve months, between 12 and 24 months and over 24 months, and organizational breadth, which can be less than a product/project team, one product/project team or multiple product/project teams.

The input parameters as well as their measure units are determined by the desired result variables and the identified process abstractions. These variables can be viewed to remain constant for the duration of the simulation, or they may vary over time. The variables in the model may be independent or have interdependencies, which can also be represented in the model. The result variables, as indicated by their name, are the variables we want to obtain from the simulation.

A series of parameters have been identified from the literature. Table 2 lists the typical elements included in process abstraction, input parameters and result variables.

3.2. Difficulties of using simulation models in software engineering

Although there are numerous benefits of using simulation models in software engineering, there are also difficulties. These difficulties are not specific to assessments in technology adoption, they can occur in all applications.

Multiple articles mention that a great amount and detail of data from the technology development processes is needed to implement a SPSM. For technology adoption, this is of particular importance because improvements are analyzed through changes in the initial

parameters. In [10], it is considered that to be able to make a software adoption assessment, a CMMi maturity level 5 is needed.

Other difficulty derives from the specialized knowledge required to use the simulation approaches. For example, it is necessary to understand the simulation approach and the simulation software package. A firm will have to invest time and resources to instruct people inside the company or integrate the experience from the outside.

Process abstraction	Input parameters	Result variables
key activities and tasks; primary objects; vital resources; activity dependencies, flows of objects among activities and sequencing; iteration loops, feedback loops and decision points; other structural interdependencies.	amount of incoming work; effort for design as a function of size; defect detection efficiency during testing and inspections; effort to code rework as a function of size and number of defects identified for correction; defect removal and injection rates during code rework; decision point outcomes; number of rework cycles; hiring rate staff; staff turnover rate; personnel capability and motivation, over time; amount and effect of training provided; resource constraints; frequency of product version releases	effort/cost; cycle time; defect level; staffing requirements over time; staff utilization rate; cost/benefit, return of investment or other economic measures; productivity; queue lengths.

Table 2. Examples of typical elements included in process abstraction, input parameters and result variables. [6]

4. System dynamics

4.1. Background

The field of system dynamics was first developed by Jay Forrester in 1950. It is grounded in the theory of nonlinear dynamics and feedback control developed in mathematics, physics, and engineering. Under this approach, a model is constructed using a defined nomenclature and principles; after that, the model is simulated using a software program.

System dynamics provides a methodology that allows integrating multiple perspectives when analyzing complex and dynamic systems [16]. To create, understand and use of system dynamics models and simulations, it is necessary to be familiar with the principles upon which the models are constructed. System dynamics utilizes a continuous approach with feedback mechanisms.

From the system perspective, a system can be classified depending on its perspective of change over time and its past actions' effects.

If the variables that describe the system change over time the system it's consider dynamic, otherwise it is static. The dynamic behavior is part of the complexities found in real life systems. Depending on how the variables change over time, a dynamic system it can be divided in continuous, discrete or combined. If the system variables change over time without breaks or irregularities, it is continuous, compared with discrete where change is presented instantaneously at separated points in time. In a continuous system, the events are not individually tracked, instead they are considered as an aggregate that can be described using differential equations. Under this approach, time changes at a constant rate.

Another principle in System Dynamics is the closed or feedback system approach. In a closed system, the future behavior of the system is influenced by its past actions, compared to an open perspective where the outputs are not influenced by the inputs. These feedback loops are what generate and control all that changes thru time [9, 17].

System dynamics does not predict the future *per se*; the accumulations in the system and its structure are what produce the system behavior [17].

4.2. System dynamics models and diagrams

To use system dynamics, a model representing the problem must be developed. A model is an abstraction of a real system [6]. In a model, it is not necessary to include all the system's elements but only those elements relevant to the problem under analysis.

The models can be used to experiments different scenarios by changing information, parameters and structure. It is possible to test 'what if' or 'tradeoff' experiment without affecting the real system because of its cost or because it is impossible to do so without altering it. These experiments allow determining how real and proposed systems perform in time [9].

In system dynamics there are different models and diagrams to describe the boundary of a model and its structure. These models and diagrams are:

- Model boundary chart - Used to list the key variables that are considered endogenous for the model (and are therefore included) and which are considered exogenous (therefore being excluded).
- Subsystem diagram - They present the major subsystems and their coupling, the boundary of the model and the level of aggregation by presenting the different organizations or agents.
- Causal loop diagram (or "causal diagram") - It is used to diagram the feedback structure of a system. It is composed by causal links among variables with arrows to describe cause and effect. One of its limitations is that it lacks the representation of the stock and flow structure necessary to keep track of system accumulations.
- Stock and flow maps - Are used to describe underlying physical structure of a system. Stocks keep track of the accumulations in terms of material, money and information.

Flows are the rates of increase of decrease in stocks. The stocks are an important part of system dynamics because they characterize the state of the system. This diagram is especially important because it is the one which is simulated by the simulation software package.

Stock and flow maps are created using the following representation:

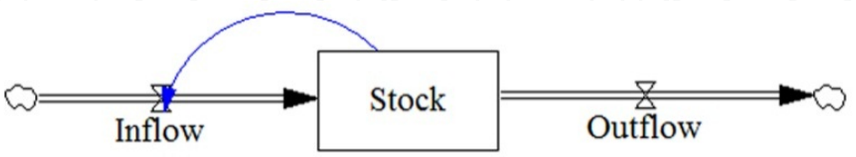


Figure 4. Basic system dynamics stock and flow map

In Table 3, each of the basic elements represented in stock and flow map are described:

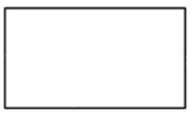
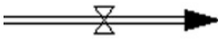
	Stock (also labeled as levels or accumulations) Known as stock, state variable or level, it's an accumulation over time. It keeps track of the accumulations of material, energy or information in the model. Stocks can only be increase or decrease by rates.
	Rates (also known as flows) These are the actions that take place in a system, controlling the flow of material, energy or information that enters and exits through the stocks.
	Source or Sink (Stocks outside model boundary) Indicate that there is an external infinite source of resources where the process modeled absolves from. The accumulation of those resources is outside of the boundary of the model.
	Information links or connections Provide data from auxiliaries or levels to rates or other auxiliaries.

Table 3. Basic elements represented in stock and flow map

Similarly to technology assessment, system dynamics uses several sources of information, such as: numerical data, written description and observations from mental models.

System dynamics models can be simulated using one of the different simulation software packages such as STELLA® [18], iThink® [19] and Vensim® [20].

http://dx.doi.org/10.5772/54498

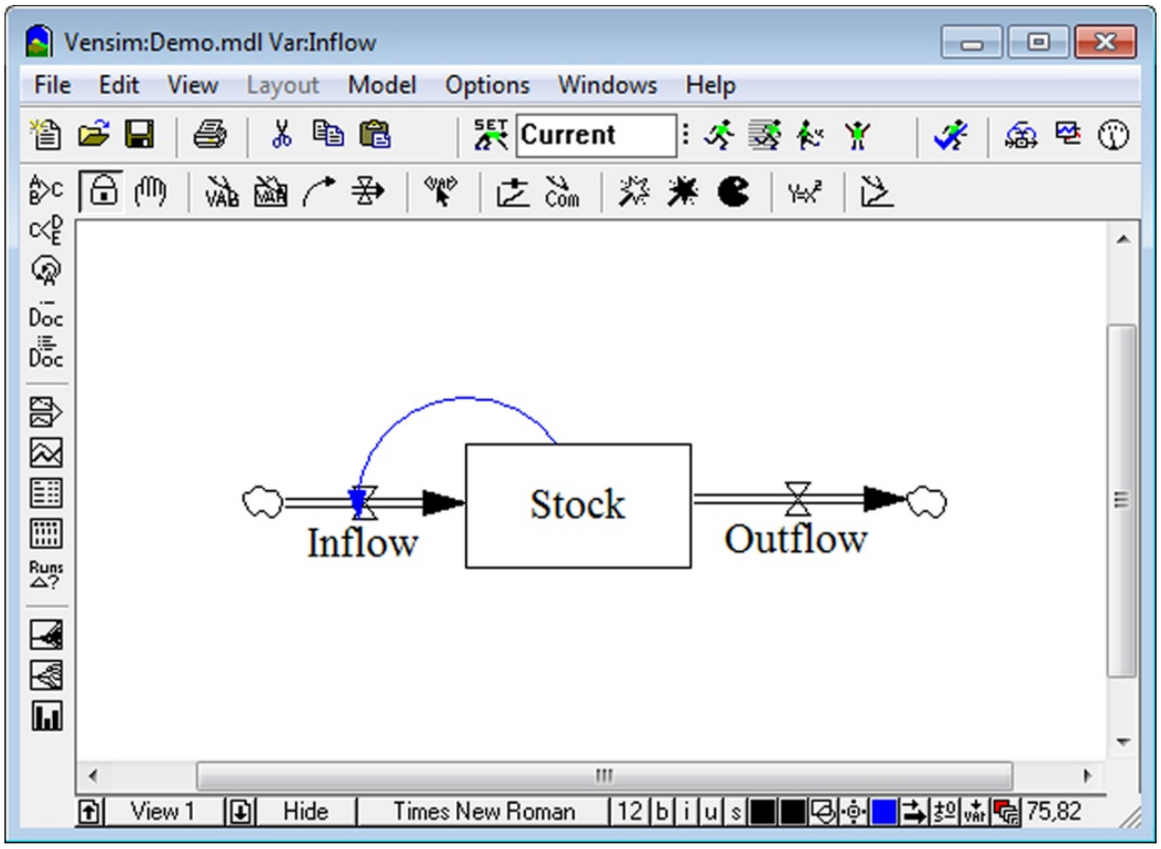


Figure 5. Screenshot from Vensim® simulation software.

5. System dynamics in the assessment of technology

There are previous examples where system dynamics has been applied in studies to evaluate the effects of technology adoption in software processes:

Fourth Generation Languages	In [21] we can find a study researching the dynamic effects of reuse and fourth generation languages in a rapid application development using a system dynamics model. The model consists of four stages: requirements, design, coding and approval phases. Different relationships between efforts are believed to exist within different programming languages. The learning curve is also deemed to vary across languages as do error rates. In the model, the reuse reduces the development effort. The study considers that models can help a project manager plan a strategy by revealing the effects of using various types of programming and their reuse levels in relation to the effort.

Commercial-of-the-Shelf components	In [22], a study analyzes the interrelation between glue code development and Commercial-of-the-Shelf components (COTS) integration. In a component based development, development of new code is needed to integrate components into the system; this is called glue code, glueware or binding code. This code is often specific to the project, and it could bring technical risks. It is difficult to decide when a software development team should start the development of glue code and integrate it into the system. One of the benefits of using the model in [22] is that it can aid in the decision of when to begin the development of glue code and integration. The model is formed by four main sections: the COTS glue code development, the COTS component factor, human resources and application development / integration. As a general rule, they found that glue code should be developed at the end of the development of the application, and the integration should occur at the beginning of the glue code development.
	In [23], the implications of specific features of a COTS-based software development process are analyzed.

Table 4. System dynamics applied to technology adoption in software development

Examples of simulation models applied to assess technologies, outside software development, include:

Defense organization	Wolstenholme, 2003: [11] The assessments of two management information systems in the defense industry are documented.
Pharmaceutical	Wolstenholme, 2003: [11] A case of a new anesthetic drug in a hospital is described.
Automotive	Yearn Min Kim, 2007: [24] Presents a forecast of the demand for in car navigation in the automotive industry; the forecast considers elements such as potential adopters, vehicle demand, navigation price, speed camera and speed fine.

Table 5. System dynamics applied to technology adoption in other fields

6. Case study

In this section, a system dynamics simulation model created to assess a reuse technology is described.

http://dx.doi.org/10.5772/54498

6.1. Application frameworks

Application frameworks is an object-oriented reuse technique [25] generally targeted to a particular application domain [16]. A framework is a "semi-complete application that can be specialized to produce custom applications" [26].

At early stages of the object-oriented field, an object was considered an appropriate abstraction level to achieve reusability, but objects were a very specialized for a particular application. Later, the concept of application framework appeared, these frameworks are more adequate to a achieve reusability in an object-oriented development [27].

Frameworks are a subsystem design composed by a collection of abstract and concrete classes with an interface between them. A framework is not a complete application by itself; it needs to be extended to create a subsystem or more specific application by instantiating specific plug-ins [27], it can be compared to a tool box [28].

Their objective is the reuse of larger-grain components and high-level designs which differ to other reuse models such as libraries [29], design patterns and components [30].

6.1.1. Framework benefits

The benefits of a framework-based development are: savings in time to market [24, 28], effort and costs, along with improvements in productivity and quality enhancement [28, 30]. These improvements are achieved as an obvious consequence of reusing code [24, 28], but also because reusing the designs that are part of the framework creates uniformity in the application [25]. Reuse of components can provide improvements in developer productivity and improve software quality, performance, reliability and interoperability [31].

According to [28], a framework "should allow you to develop the application quickly and easily and should result in a superior finished application".

6.1.2. Framework difficulties

Application frameworks also have negative sides. According to [27], application frameworks are a good approach for taking advantage of reuse but they have a high cost when they are introduced in software development processes and projects. Additionally, they create a dependency on the developed applications [25], which can affect the software maintenance.

An application framework is itself a complex software, and in order to use it, it's first necessary to understand it, which consumes software engineers' time, the learning curve can even take months [27]. This learning curve can be a decisive factor in making a framework technology profitable [28]. It is important to estimate if implementing the technology in a project or organization would be a viable alternative.

6.2. Problem statement

Application framework technologies provide benefits but add difficulties to development projects. The main benefits described in literature arise from reuse, while the main difficulty is the learning curve; in this chapter we explore the effects of the latter. As mentioned before, framework learning can be a decisive factor in making this technology profitable in organizations and projects.

6.3. Application area

The presented model can be used to analyze the adoption of an application framework technology in software development projects.

6.4. Model overview

Given that an application framework allows the reuse of code and patterns via code, and main benefits and difficulties of the framework technology occur during the software coding phase, it was defined that this phase was the domain of the technology. The model is focused on the use of the technology, not on its creation. Elements from other models described in system dynamics literature and main elements in framework-based software development were included.

The model is composed of four interrelated subsections: software development, human resources, projects and framework learning.

The human resource section, presented in Figure 6, was modeled using a subsection from the model in [13]. This section allows simulating different scenarios for active developers in a project. Two categories of developers are considered: beginners and experts. The first category takes a predefined period of time to turn into expert. Both categories have a potential productive capacity.

There are two main issues affecting potential productive capacity. The first is that the expert programmers will be asked by beginners to help resolve questions and problems, absorbing productive capacity. Furthermore, a loss in productivity is considered due to communication between team members. The more team members, the more time is consumed in communication exchanges.

Given that in the literature it is exposed that becoming proficient at a framework technology can take up to several projects, the model includes a section called Projects. This section allows simulating the development of more than one consecutive projects. A new project will begin as soon as the previous one is completed, until the set number of projects has been completed. This section is presented in Figure 7.

Because the model is presented by segments, the relationship between the “In progress” rate from the project section and the “Project artifacts” rate from the Software development section was omitted.

http://dx.doi.org/10.5772/54498

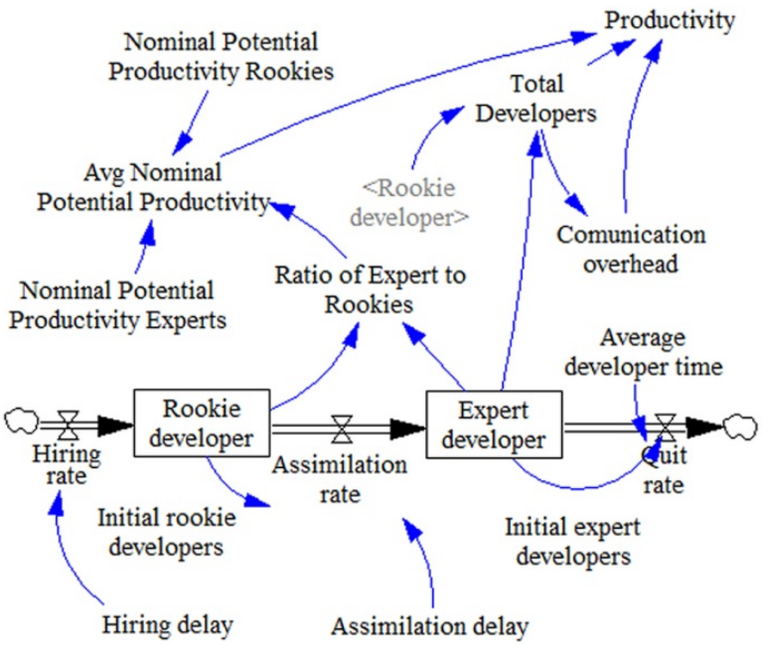


Figure 6. Human resource subsection

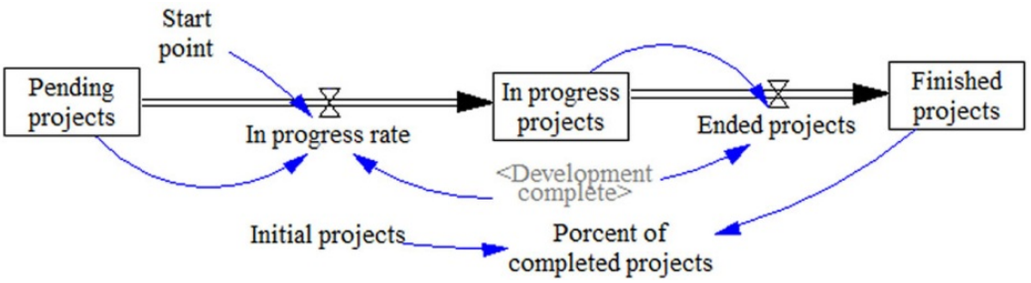


Figure 7. Projects section

The software development section, shown in Figure 8, derives from different models where software development is simulated. In the model, the term artifact is used to not limit the measure unit to be simulated, for example: function points, object oriented function points, code lines, etc. Similar to the model in [13], a productivity multiplier is included as a result of learning from the project. The incidence of defects that will have to be rebuilt and might affect the conclusion of the project is also taken into account.

The framework learning subsection represents the learning of the technology. As mentioned before, framework learning is a decisive factor to consider when introducing this technology. For this reason, this subsection will be described in detail.

The learning curve is an element which frequently appears to be a difficulty in the implementation of new technologies in software development projects. The role played by this element is fundamental. Among other consequences, it can affect programmer productivity

and increase difficulty in maintenance. Since the learning curve varies over time in a project, it can be analyzed in detail using simulations in the assessment of a technology.

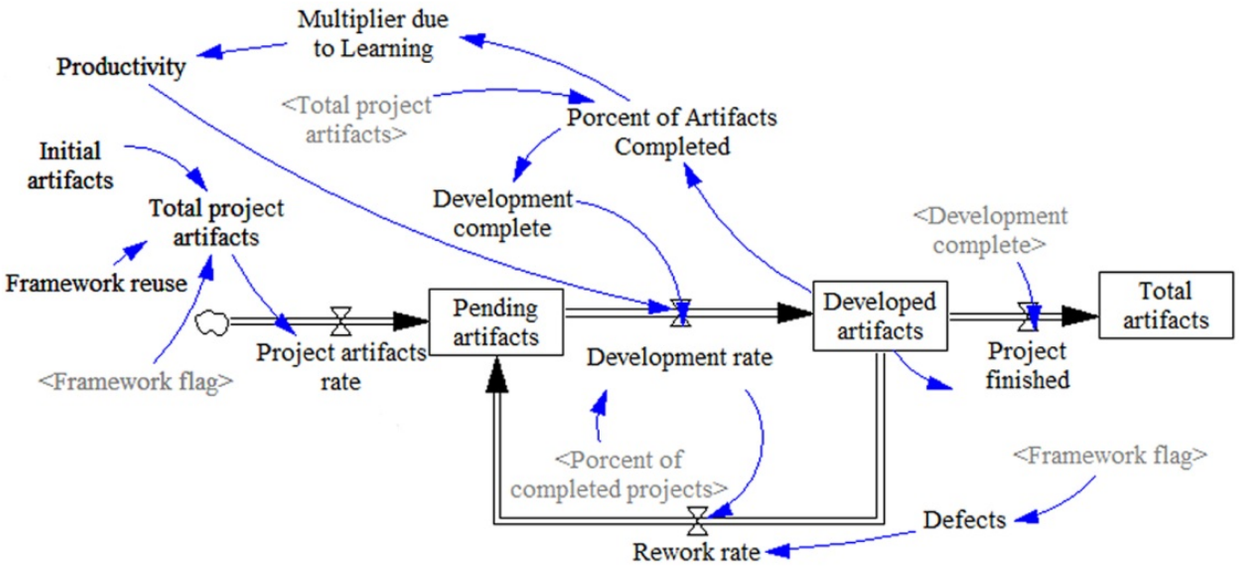


Figure 8. Software development section

The model considers that the programmer can learn about the framework while developing artifacts in a project. This knowledge is lost if the programmer quits the project. The knowledge left over after a programmer leaves depends on the amount of programmers working in the project. If the sole programmer in a project leaves, all knowledge is lost. If only one programmer in a bigger team leaves, one part of the knowledge is lost, while other parts have been transferred and distributed to other team members.

In the model, it is also considered that programmers begin with no framework knowledge. Nonetheless, this section enables to run simulations where developers start with varying knowledge levels due to education or training. The framework learning subsection described before is presented in Figure 9.

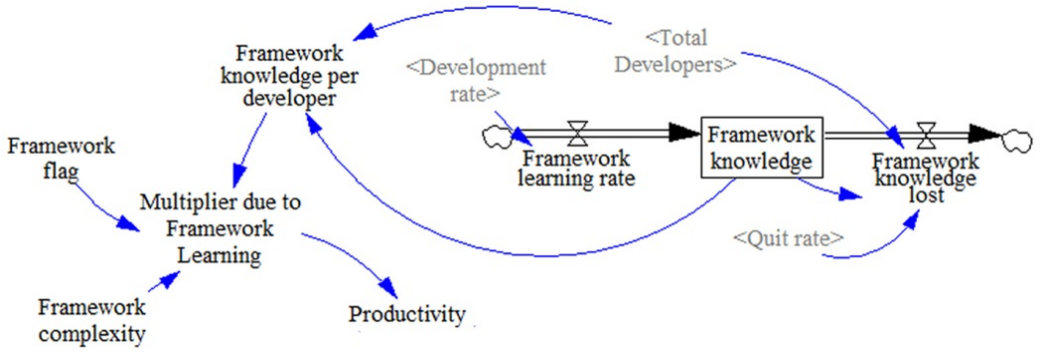


Figure 9. Framework learning subsection

The framework learning subsection is composed by four main elements:

Framework knowledge	In the model, the programmer gains framework knowledge by developing artifacts. Framework knowledge is the accumulation of artifacts developed by the programmers. It is the sum of all the framework knowledge of the developers.
Framework knowledge per developer	This variable is used to calculate the average knowledge per developer.
Framework complexity	The framework complexity variable is used to indicate how many artifacts a programmer must develop to reach the maximum expected benefits in productivity as a result of the framework usage.
Multiplier due to Framework Learning	This variable is the application framework learning curve. This learning curve generates an improvement or decline in developer productivity as a result of framework knowledge. The position in the learning curve is calculated based on framework complexity and the number of artifacts built with the framework per developer.

Table 6. Main elements framework learning subsection

6.5. Simulation model calibration and results

This section presents the main data used to configure the model and the results obtained from the simulations. The software package Vensim® was selected to generate the simulations.

The research data in [29] was used as a base to configure the model simulations. In [29], the authors made an exploratory case study where one subject developed five applications using a framework and four applications without one. These nine applications were developed in a research and development company in a six month period using a proprietary framework. The findings show that the productivity and quality is increased in both stances, however, using a framework to develop applications increased productivity more significantly.

In the present study, the model will use the following configuration settings:

- The simulated artifacts will be based on Object Oriented Function Points (OOFP).
- The simulation will comprise five projects
- The initial number of expert programmers is set to one and the amount of beginner programmers is cero.
- The average project size will be 1370 OOFP. After a decrease of 80% due to framework reuse, the average project size will be 274 OOFP.
- Nominal potential productivity of expert programmers will be established at 1 OOFP/Hour.
- A 10% defect percentage will be considered.

- To reach the plateau of the learning curve, the subject must develop 1370 OOFP using the framework.

The data presented above remained unchanged in the presented simulations with the purpose of identifying the impacts or effects of an isolated element, the learning curve. In this study the impacts of two different application framework learning curves are explored. In scenario 1, the curve is generated from the data in [29]. In the cited study, productivity was not found to decrease in the technology learning period; instead, during the five projects, productivity was shown to increase. As a result, productivity in scenario 1 is not lowered by the learning and use of the framework.

A similar curve is used in scenario 2, but productivity is lower until 20% of framework knowledge is achieved. Once the 20% knowledge has been reached, the negative effect in productivity is overcome, but productivity is still lower than in scenario 1 until framework knowledge reaches 42%. From this point forward, both curves are equal. Figure 10 exhibits the two learning curves (Scenario 1 and Scenario 2) used in the model. The curves are represented based on their effect in programmer productivity.

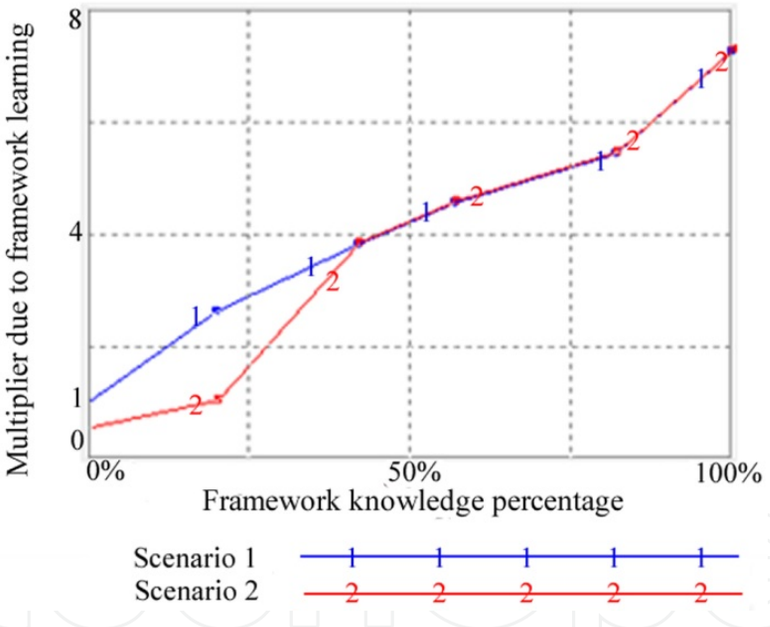


Figure 10. Learning curves used in the model.

Two simulations, one for each scenario described before, were executed. The results of the simulations are presented in Figure 11 and Table 7. Figure 11 was generated using the Vensim® report functionalities.

In Figure 11, the amount of OOFP developed in time is presented. Each of the peaks in the blue line (Scenario 1) or the red line (Scenario 2) corresponds to the conclusion of each simulated project. The x-axis represents time measured in hours; as it increases, the amount of developed artifacts in each project is increased in the y-axis. Once all artifacts, in this case

http://dx.doi.org/10.5772/54498

OOFP, in a project have been developed, the amount of developed artifacts is reset to cero to begin a new project.

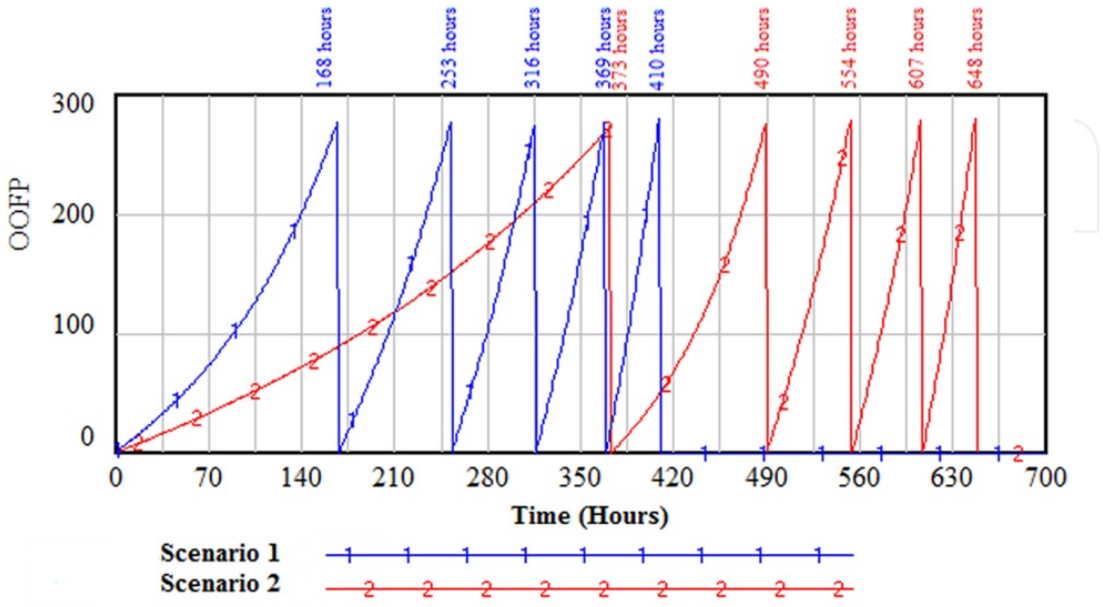


Figure 11. Results of the simulation – Developed artifacts

Table 7 lists the hour in which each of the five projects was concluded in sequence by scenario. In scenario 1, the five simulated projects end after 410 hours, while in scenario 2 they end after 648 hours, a difference of 238 hours. The data show a 58% percent increase in scenario 2 compared to scenario 1. In other words, projects in scenario 2 would end 58% later than in scenario 1.

	Time of conclusion	
Project number	**Scenario 1**	**Scenario 2**
1	168 hours	373 hours
2	253 hours	490 hours
3	316 hours	554 hours
4	369 hours	607 hours
5	410 hours	648 hours

Table 7. Time of conclusion for each simulated project

Results show that the learning curve has an important effect in development time because of its relationship to productivity. Considering that learning is achieved through project development, this impact in productivity can consequently inhibit framework learning.

7. Conclusions

7.1. General conclusions

The accelerated rhythm of technology advancements pressures managers and organizations to evaluate what is the best alternative for their particular needs. Deciding which technologies to implement in software projects and processes is a challenging task, but of paramount importance.

The area of technology assessment provides decision makers with guidelines to achieve this task; it relies on different techniques to analyze the impact of these technologies and the aspects involved. Simulation modeling techniques, such as system dynamics, make it possible to analyze the impact of adopting a technology before its implementation.

System dynamics has been applied in the assessment of new technology adoption in the field of software development. It offers the unique advantage of being able to include a variety of perspectives and information sources, including qualitative and quantitative data. This technique also allows isolating particular elements for the purpose of impact analysis, as in the case study presented. This is significantly beneficial in the study of complex system with multiple interrelated elements and can provide meaningful insights.

The area of model simulation in software has a broad range of applications, from support of strategic decisions to project planning and also, as described in this chapter, technology adoption.

7.2. Case conclusions

A model to evaluate the use of frameworks in software development projects was presented in the chapter. The learning curve is an important aspect to consider in the development and use of frameworks because it can make it unviable to use a technology in a project or organization. For this reason, this element was selected to be isolated and studied using system dynamic simulations.

Two scenarios with different learning curves were simulated to analyze the effect of framework learning. The changes in the learning curve had a significant effect in the time of conclusion of a series of sequential projects. We consider that the results of the simulations presented concur with the literature, which is a positive finding, given that the model aims to represent as best as possible the reality of using framework technologies in projects.

8. Future research

Various research opportunities can be developed from this study; the structure of the model presented for application frameworks will continue to be researched, experimenting with different configurations and initial parameters. The importance of the learning curve for this technology will be analyzed, through the use of more model simulations. As for the relation

between technology assessment and software process simulation models, it is possible to make a systematic literature review from the perspective of technology adoption.

Author details

José Ignacio Muñoz Hernández[1*], José Ramón Otegui Olaso[2] and Alejandro Gutiérrez López[2]

*Address all correspondence to: JoseIgnacio.Munoz@uclm.es

1 University of Castilla-La Mancha UCLM, Spain

2 University of the Basque Country UPV/EHU, Spain

References

[1] Ernest B. Technology in Context: Technology Assessment for Managers. 1st ed. London: Taylor & Francis; 1998.

[2] Torkkeli M, Tuominen M. The contribution of technology selection to core competencies. Int J Prod Econ 2002 6/11;77(3):271-284.

[3] Daim TU, Intarode N. A framework for technology assessment: Case of a Thai building material manufacturer. Energy for Sustainable Development 2009 12;13(4): 280-286. http://www.sciencedirect.com/science/article/pii/S0973082609000751 (accessed 6 August 2011).

[4] Sterman JD. Business Dynamics: Systems Thinking and Modeling for a Complex World. United States of America: McGraw-Hill; 2000.

[5] Pfahl D, Lebsanft K. Using simulation to analyse the impact of software requirement volatility on project performance. Information and Software Technology 2000;42(14): 1001-1008. http://linkinghub.elsevier.com/retrieve/pii/S095058490000152X (accessed 11 November 2010).

[6] Kellner MI, Madachy RJ, Raffo DM. Software process simulation modeling: Why? What? How? J Syst Software 1999;46(2-3):91-105. http://linkinghub.elsevier.com/retrieve/pii/S0164121299000035 (accessed 13 June 2011).

[7] Van Den Ende J, Mulder K, Knot M, Moors E, Vergragt P. Traditional and Modern Technology Assessment: Toward a Toolkit. Technological Forecasting and Social Change 1998 0;58(1–2):5-21. http://linkinghub.elsevier.com/retrieve/pii/S0040162597000528 (accessed 3 August 2012).

[8] Henriksen ADP. A technology assessment primer for management of technology. Int J Technol Manage 1997;13(5/6):615-638. http://inderscience.metapress.com/content/yu9yq058feyd6avh/ (accessed 12 August 2012).

[9] Madachy RJ. Software Process Dynamics. 1st ed.: Wiley-IEEE Press; 2008.

[10] Raffo DM, Wakeland W. Moving Up the CMMI Capability and Maturity Levels Using Simulation. Pittsburgh: Software Engineering Insitute, Carnegie Mellon University; 2008. http://www.sei.cmu.edu/library/abstracts/reports/08tr002.cfm (accessed 3 September 2012).

[11] Wolstenholme EF. The use of system dynamics as a tool for intermediate level technology evaluation: three case studies. J Eng Technol Manage 2003;20(3):193. http://linkinghub.elsevier.com/retrieve/pii/S0923474803000183 (accessed 21 October 2010).

[12] Zhang H, Kitchenham B, Pfahl D. Software process simulation modeling: Facts, trends, and directions. Proceedings of 15th Asia-Pacific Software Engineering Conference, APSEC 08, 2-5 December 2008, Beijing, China.

[13] Abdel-Hamid T, Madnick S. Software Project Dynamics: An Integrated Approach. Prentice Hall; 1991.

[14] Paulk MC, Weber CV, Garcia SM, Chrissis MB, Bush M. Key Practices of the Capability Maturity Model Version 1.1. Software Engineering Institute 1993;Paper 171. http://www.sei.cmu.edu/reports/93tr025.pdf (accessed 3 September 2012).

[15] Münch J, Armbrust O, Kowalczyk M, Soto M. Software Process Simulation. In: Münch J, Armbrust O, Kowalczyk M, Soto M (eds.) Software Process Definition and Management. Berlin Heidelberg, Springer; 2012. p187-210. Available from http://www.springerlink.com/content/xv1p58082762075w/ (accessed 3 September 2012).

[16] Fayad ME, Schmidt DC, Johnson R. Implementing Application Frameworks: Object-Oriented Frameworks at Work. John Wiley & Sons; 1999.

[17] PhD seminar in system dynamics conducted by Jay W. Forrester at the MIT Sloan School of Management [DVD]. Albany, NY1: Sloan School of Management. System Dynamics Group; 1999.

[18] isee systems, inc. STELLA Systems Thinking for Education and Research. http://www.iseesystems.com/softwares/Education/StellaSoftware.aspx (accessed 3 September 2012).

[19] isee systems, inc. iThink Systems Thinking for Business. http://www.iseesystems.com/softwares/Business/ithinkSoftware.aspx (accessed 3 September 2012).

[20] Ventana Systems, Inc. Vensim Software. http://www.vensim.com/software.html (accessed 3 September 2012)

[21] Lo KW. Reuse and high level languages. Center for Systems and Software Engineering, University of Southern California 1999; CS599 Final Report. http://sunset.usc.edu/classes/cs599_99/projects/reuse.pdf (accessed 3 September 2012).

[22] Kim WK, Baik J. Dynamic model for COTS glue code development and COTS integration. Center for Systems and Software Engineering, University of Southern California 1999; CS599 Final Report. http://sunset.usc.edu/classes/cs599_99/projects/COTS.pdf (accessed 3 September 2012).

[23] Ruiz M, Ramos I, Toro M. Using Dynamic Modeling and Simulation to Improve the COTS Software Process. Lecture Notes in Computer Science 2004;3009:568-581. http://www.springerlink.com/index/MDEF683VQ3LKHGNB.pdf (accessed 12 September 2011).

[24] Kim YM. A system dynamics model for the technological forecasting of automotive in-car navigation market. Proceedings of Second Intl Forum on Strategic Technology, IFOST 2007, 3-6 October 2007, Ulaanbaatar, Mongolia.

[25] Johnson RE. Frameworks = (components + patterns). Commun ACM 1997;40(10): 39-42. http://dl.acm.org/citation.cfm?doid=262793.262799 (accessed 25 October 2011).

[26] Johnson RE, Foote B. Designing Reusable Classes. J Object-Oriented Programming 1988;1(2):22-35.

[27] Sommerville I. Ingeniería de Software. 7th ed. Madrid: Pearson Educación S.A.; 2005.

[28] Nash M. Java Frameworks and Components: Accelerate Your Web Application Development. Cambridge University Press; 2003.

[29] Morisio M, Romano D, Stamelos I. Quality, Productivity, and Learning in Framework-Based Development: An Exploratory Case Study. IEEE Trans.Softw.Eng. 2002;28(9):876-888. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1033227 (accessed 12 September 2011).

[30] Polancic G, Horvat RV, Rozman I. Improving Object-Oriented Frameworks by Considering the Characteristics of Constituent Elements. J Inf Sci Eng 2009;25(4): 1067-1085. http://www.iis.sinica.edu.tw/page/jise/2009/200907_07.pdf (accessed 12 September 2011).

[31] Fayad M, Schmidt DC. Object-oriented application frameworks. Commun ACM 1997 October;40(10):32-38. http://dl.acm.org/citation.cfm?doid=262793.262798 (accessed 22 June 2011).

IntechOpen

IntechOpen