THE IMPLEMENTATION OF APPLICATION FRAMEWORKS IN SOFTWARE DEVELOPMENT PROJECTS: A SYSTEM DYNAMICS MODEL

Gutiérrez López, A. [1]; Otegui Olaso, J. R.[1]; Muñoz Hernández, J. I.[2]

[1] Universidad del País Vasco, [2] Universidad de Castilla-La Mancha,

Technology selection is a critical decision in projects and organizations. In the software industry, reuse technologies aim to reduce projects costs and duration. Among these technologies, application frameworks or frameworks stand out due to their potential to reuse high level designs and components via code.

Before introducing a framework, it is necessary to analyze its expected benefits and difficulties. Once the coding phase in a development project has been initiated with a framework, it is not possible to eliminate it, unless the development is restarted. System dynamics simulation models make it possible to analyze the impacts of a new technology before its implementation; furthermore, it is possible to analyze the selection not only in terms of the technology itself, but also in how the decision could affect other aspects of the project.

The objective of this article is to study the effects of implementing a framework using system dynamics model simulations. Using the proposed model it is possible to analyze the expected benefits and difficulties from this technology under different scenarios.

Keywords: *Technology selection; System dynamics; Software development; Project management; Application frameworks*

THE IMPLEMENTATION OF APPLICATION FRAMEWORKS IN SOFTWARE DEVELOPMENT PROJECTS: A SYSTEM DYNAMICS MODEL

La selección de la tecnología adecuada es una decisión crítica en los proyectos. En la industria del software, tal decisión afecta a la incorporación de tecnologías de reutilización que buscan reducir los costos y duración de los proyectos. Entre estas tecnologías, los marcos de trabajo destacan por su capacidad de reutilizar diseños de alto nivel y componentes.

Antes de introducir un marco de trabajo, es necesario analizar sus beneficios – reducción de errores; mayor velocidad de codificación - y sus desventajas - necesidad de aprendizaje; vinculación permanente con la aplicación desarrollada-.

Una forma de analizar el impacto de la introducción de marcos de trabajo puede ser la dinámica de sistemas. Durante este proyecto de investigación, se utiliza esta herramienta para la simulación previa a la implementación. Los resultados consolidan investigaciones previas, y resaltan la importancia de la curva de aprendizaje.

Palabras clave: *Selección de tecnologías; Dinámica de Sistemas; Desarrollo de software; Dirección de proyectos; Marcos de trabajo*

Correspondencia: Dpto. de Expresión Gráfica y Proyectos de Ingeniería. Alameda de Urquijo s/n. C.P. 48013. Bilbao, Bizkaia, España.

1. Introduction

New technologies appear continuously with the purpose of improving software development. Nowadays, software architects, developers and project managers can choose from a plethora of options in tools, methods and techniques. Selecting a potentially viable technology alternative can be a strenuous task for each organization or project.

To assess the possible impacts of the adoption of a new technology one option is to design and execute experiments. However, the experimentation with alternative technologies in real projects is considered costly in terms of time and money (Pfahl and Lebsanft, 2000).

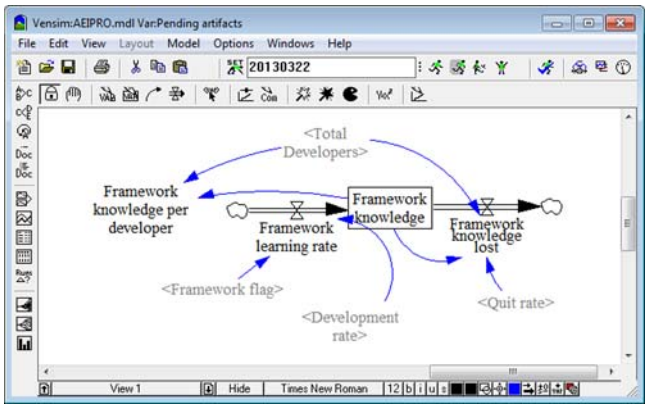
Another option is to make use of simulation models to assess the implementation of new technologies before they occur. A simulation model is a computerized representation of a real or conceptual system (Kellner, Madachy and Raffo, 1999); it shows the system's features and characteristics so they can be studied, predicted, modified or controlled. Building and simulating models is a feasible option whenever experimenting on the real system is not (Pfahl and Lebsanft, 2000). Simulations models are created using software such as the one shown in Figure 1.

Through simulation models, one can analyze the introduction of a technology considering the complex relations of the system where it would be implanted.

The possibility of using simulations to assess technologies has been described by different authors (Wolstenholme, 2003; Madachy, 2008; Raffo and Wakeland, 2008); in the software field, Raffo and Wakeland (2008) use process simulation models to assess a requirement analysis tool. Madachy (2008) describes a model to assess the use of third and fourth generation programming languages, and he also established that the use of simulations can help organizations to evaluate their technology strategies.

In this article, a simulation model is used to analyze the implementation of a reuse technology called *application frameworks* or *frameworks* in a development project. Application framework technologies aim to reuse high level design and components in software development projects.

Figure 1: VENSIM® System Dynamics Simulator



The following sections will state the purpose of this study, explain the framework's main characteristics and the methodology applied. The system dynamics model used to assess the technology will be presented next, as well as the results of the simulations and conclusions.

2. Objective

The purpose of the research presented in this paper is to examine the effects of different S-shaped framework learning curves on the completion time of the coding phase in a small software development project, by means of a system dynamics simulation model.

In this document we describe the application framework technology and a methodology to evaluate technologies using system dynamics models. The elements of the model used to represent framework learning are presented along with the simulation results. In the simulation, the framework learning curves are generated applying the Monte Carlo method.

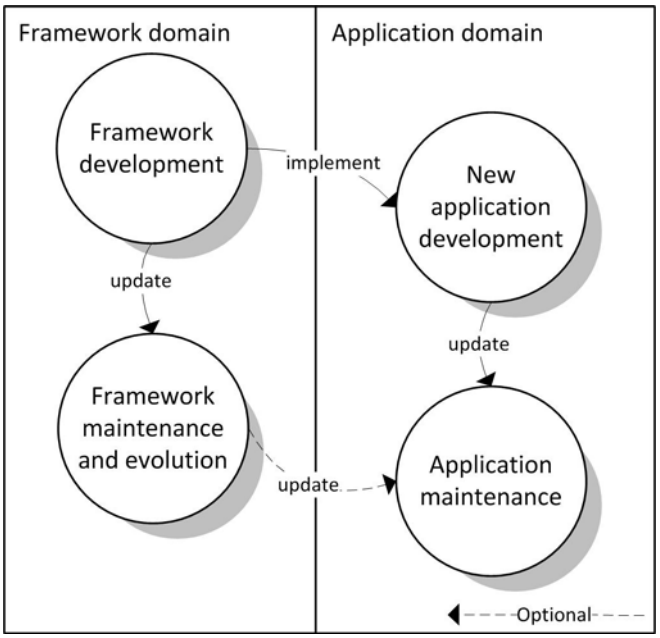
3. Application frameworks

From the dawn of software development, the discipline has searched for reuse (Morisio, Romano and Stamelos, 2002). Application frameworks have appeared as the result of object oriented programming and the languages grounded on this paradigm. At the beginning of object oriented programming, objects were thought to be an appropriate abstraction level for reuse, but the objects were specialized for a singular application (Sommerville, 2005). Later, the concept of framework appeared as a more suitable reuse mechanism in object oriented programming processes (Sommerville, 2005).

A framework is a subsystem composed by a collection of abstract and concrete classes with an interface between them (Sommerville, 2005). It is not a complete application by itself, it needs to be extended to create a subsystem or to create a more specific application by instantiating particular connectors (Sommerville, 2005). The purpose is to reuse high level designs and components by the means of code (Morisio, Romano and Stamelos, 2002).

Four main activities in framework based developments can be identified: first, the building of the framework; second, the development of an application using the framework; third the maintenance and evolution of the framework and last, the maintenance of the application.

Figure 2: Framework based developments



First, the development of the technology requires a great amount of hours, and much more resources than the construction of a traditional application (Morisio, Romano and Stamelos,

2002). The difficulties and challenges of building a framework stem from the fact that the framework must allow to be reused in multiple applications.

Once the framework is ready, the framework is implemented in a new software application. It use can bring benefits such as increase in productivity and reduction in the amount of errors (Fayad and Schmidt, 1997). This, in turn, can decrease the time and cost of development and increase software quality. To some extent, these benefits are achieved as a result of the time saved in development because of the reutilized code in the framework.

Regarding their difficulties during use, different authors have described the framework learning curve as one of the biggest challenges (Fayad and Schmidt ,1997; Morisio, Romano and Stamelos, 2002). In certain projects, this curve may well determine whether the technology is feasible to use.

Fayad and Schmidt (1997) posit that in some cases, the effort of learning how to use a framework will have to be absorbed among several projects. It is also necessary to consider that developers will normally require training and mentoring to use frameworks adequately (Fayad and Schmidt, 1997).

Furthermore, because of the reuse of high level designs, the application being developed must accept the control flow defined by the framework. This change in paradigm is known as “inversion of control”, because the flow was traditionally defined by the application (Morisio, Romano and Stamelos, 2002); a framework can substantively change the way to develop an application.

Additionally, the framework maintenance and evolution and the upkeep of the applications developed with it needs to be considered. On one hand, application requirements change constantly, which can cause that the framework requirements change as well (Fayad and Schmidt, 1997). By contrast, the framework could evolve, and decisions must be made whether to update the applications developed with it. In both cases, this will translate into efforts that must be considered for the projects. In either case, it will be necessary to consider the effort this will mean for the projects.

Before embarking on a project using frameworks, it must be understood that the relationship between the framework and the application will last for the entire lifecycle of the application; the application cannot exist without the framework unless it is built anew.

4. Methodology

This research has followed the three stage methodology proposed by Wolstenholme (2003) for the use of system dynamic models to evaluate a technology. The stages are described in Table 1.

Table 1: Three stage methodology by Wolstenholme

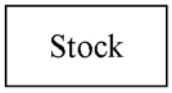
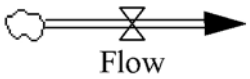
Stage I Model the domain of application	Creation of a system dynamics model that includes the domain area of the technology to evaluate. The development of the model is based on the operation as is before the technology implementation.
Stage II Technology assessment	The model built in Stage I is used to test the impacts of the new technology. Changes expected from the implementation of the new technology are incorporated to the model built previously.
Stage III Technology accommodation in the domain	To get maximum benefits from the new technology, possible adjustments, changes or optimal levels are analyzed using the model. Improvements, among others, can include: process redesign, changes in the organization's operative limits, increase or decrease in capabilities.

Wolstenholme (2003) describes these evaluations through system dynamics as an intermediate level technology assessment; positioned halfway between the perspective of an economic analysis and a detailed analysis of the characteristics and features of the technology.

With the system dynamics models, it is possible to consider the characteristics of the technology whilst also providing an evaluation that includes its application domain.

System Dynamics is a perspective and collection of tools (diagrams, simulation software, among others) used with the objective to understand complex systems, their structure and dynamics (Sterman, 2000). From the system dynamics diagrams, this research will make use of the Stock and Flow diagram. The Stock and Flow diagram is used to represent the structure of the system and simulate it quantitatively; and it holds a specific notation to build the model, which is shown in Table 2.

Table 2: Stock and flow diagraming notation

	Stock: They simulate the accumulation of matter, energy or knowledge in a given moment in the system. A stock is used for each type of matter, energy or knowledge to be simulated.
	Flow: It determines the amount of incoming and outgoing matter, energy or knowledge from the stocks in the system.
Auxiliary	Auxiliary variable: These variables provide a clearer sense to the relations within the model.
	Information flow: They link the stocks, rates and variables that relate in the model.

Additionally, this research will generate different framework learning curves by applying Monte Carlo simulations. Using Monte Carlo simulations is possible to define starting values as distributions of probability.

5. Case study

The case of an application framework evaluation for a software development project is described next.

5.1. Stage I - Model the domain of application

To assess technologies through simulations, it is first necessary to build a model representing the domain where the technology is to be implemented.

The responsibility of implementing a framework is set largely on the software developer, which is why the project coding phase was considered to evaluate the technology. The coding phase is essentially a developer or group of developers that turn a number of requirements, specifications and descriptions into code at a certain rate or productivity.

Based on research by Abdel-Hamid and Madnick (1991) it is considered that knowledge on the project is increased along with the progress of developers on development activities. As a result, productivity is also increased.

5.2. Stage II - Technology assessment

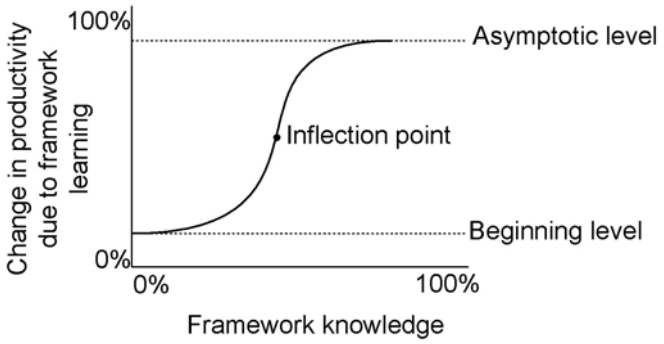
The changes expected to be brought on by the new technology are introduced in the second stage of the applied methodology.

The greatest benefit of using frameworks is the decrease in the amount of code needed to be developed, so the amount of code to be developed should be reduced in order to simulate different reuse levels.

On the other hand, an element that appears in the literature as part of the difficulties of implementing a framework is its learning curve. The learning curve plays a fundamental role because it can slow down developer productivity and delay the completion of the project.

This research considers that the framework learning curve will follow a symmetric S-shape, shown in Figure 3.

Figure 3: S-shaped learning curve



The framework learning curve will be represented using a logistic growth function; this function is composed by four constants: beginning level, asymptotic level, slope and inflection point. The initial level represents a performance level before having experience or skills in the task to be done; in this case, the development of software using a framework. The asymptotic level is the maximum performance level to be achieved, assuming that the environment or the factors that impact performance do not change. The slope is the rate of change due to learning and the inflection point is where the acceleration of the curve changes.

Figure 4: Model used to simulate the framework learning

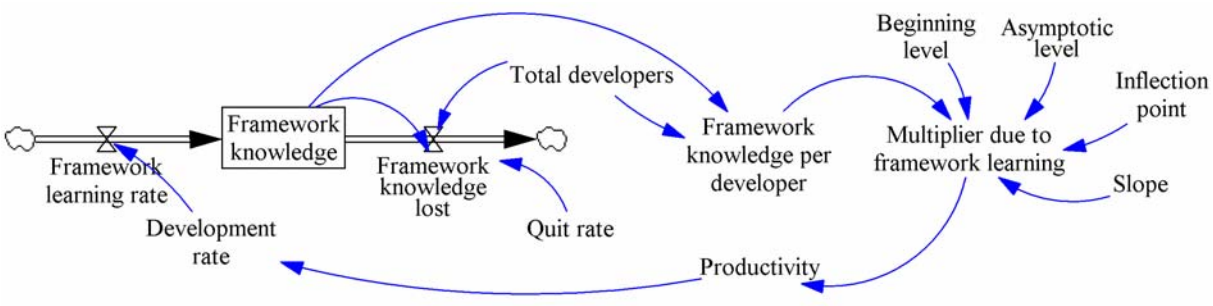
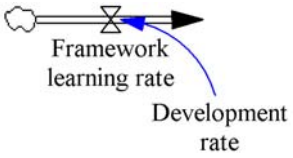
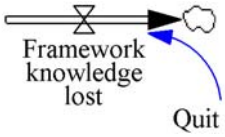
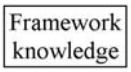


Figure 4 shows the model used to simulate the framework learning curve and the framework knowledge acquired by the developers; additionally, Table 3 describes the elements of the model in detail.

Table 3: Framework learning model description

	The model simulates the developer learning about the framework in the coding stage. As the developer progresses in the development, the framework learning is increased.
	When one developer leaves the project, he/she takes part of the total knowledge acquired about the framework.
	This variable indicates the level of knowledge about the framework in the project. It is the sum of all the knowledge about the framework acquired by the developers in the project.
Total developers	Amount of active developers in the project.
Framework knowledge per developer	Average framework knowledge per active developer in the project.
Multiplier due to framework learning	Determines the effect of the learning curve on developer productivity, based on the framework learning curve and the level of technology knowledge at a given moment of the simulation.
Beginning level Asymptotic level Slope Inflection point	These elements shape the learning curve into an S. In this case, they are used to shape the framework learning curve.
Productivity	Coding rate for active developers. Productivity impacts the development rate directly, the greater the productivity, the greater the development rate.
Development rate	Turns pending coding lines (to be developed) into coded lines at the rate established by the productivity.

5.3. Stage III - Technology accommodation in the domain

Once the expected changes have been introduced into the model, it is possible to explore changes in operative capacity in the elements to find an optimal level. The effects of different framework learning curves on completion of the coding stage in a project are analyzed in this research.

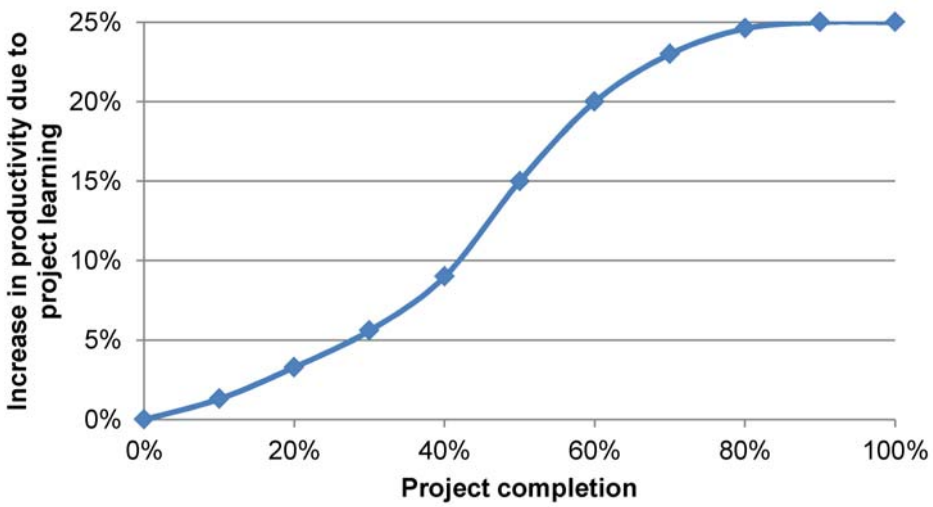
To start, it is necessary to define the scenario to be simulated. In this research, the simulated scenario is a project with a size of 5000 source lines of code (SLOC).

This research will not study the effects of different reuse levels; thus, it will be assumed that the work to be made (5000 SLOC) is the result of having reused code through a framework.

The amount of source lines of code will be made by a developer at an initial productivity rate of 20 SLOC/Hour.

As described in section 5.1, the progress on development activities augments the project learning and the productivity. In the simulations, the improvement in productivity will be set based on Abdel-Hamid and Madnick's (1991), Figure 5 shows the values.

Figure 5: Increase in productivity due to project learning



Previous data will remain constant in the simulations to identify the effects of one isolated element, the framework learning curve.

Assuming that the framework learning curve parameters are unknown, Monte Carlo simulations will be used to generate different learning curves to be able to determine the results. Possible value ranges must be defined in Monte Carlo simulations. This research considers that the initial framework learning curve will impact initial productivity with a decrease of 40 to 60%.

Once the developer has acquired knowledge on the framework, the productivity will reach up to 110%. This represents that after learning the framework will increase the productivity up to 10% more.

For the slope, the values considered a range from 300 to 500. The inflection point is calculated based on the percentage of initial lines of code, which is 40 to 60% for the simulated scenarios or, converted to initial project lines of code, 2000 to 3000 SLOC. Using

the abovementioned ranges, 2000 simulations will be made, all of them using a uniform probabilistic distribution.

6. Results

Figures 6 and 7 show the results obtained from the simulations. Figure 6 represents the different learning curves simulated in the research and Figure 7 shows the coding phase completion time. Both images are shown as confidence bounds for different percentages (50%, 75%, 95%, and 100%).

Figure 6: Simulated framework learning curves

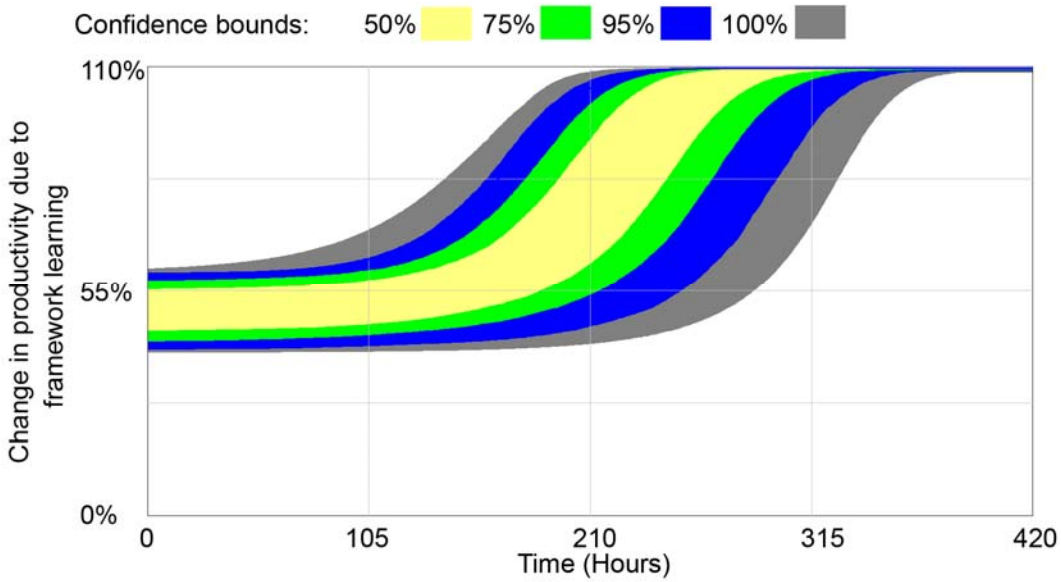
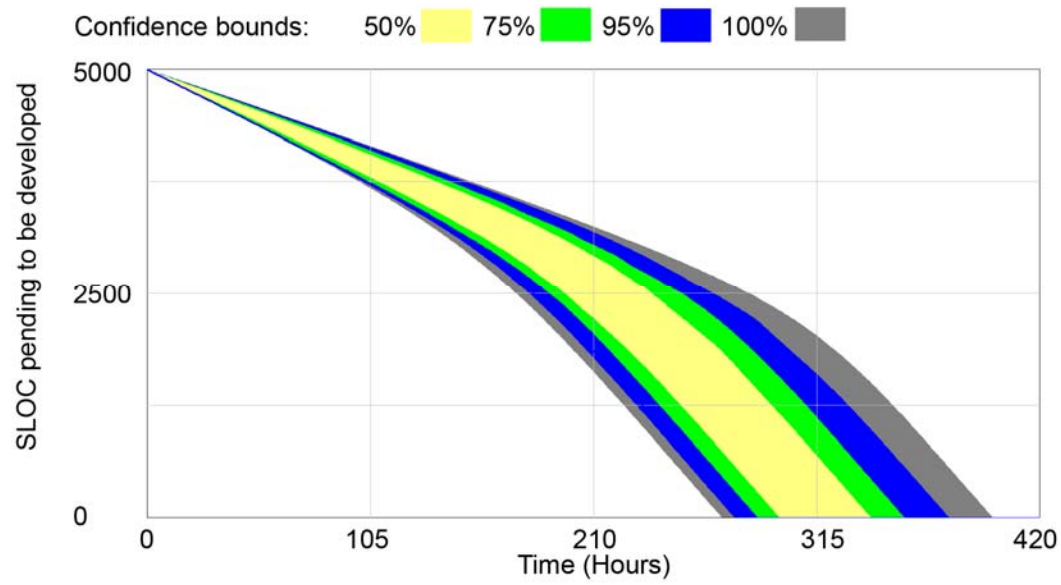
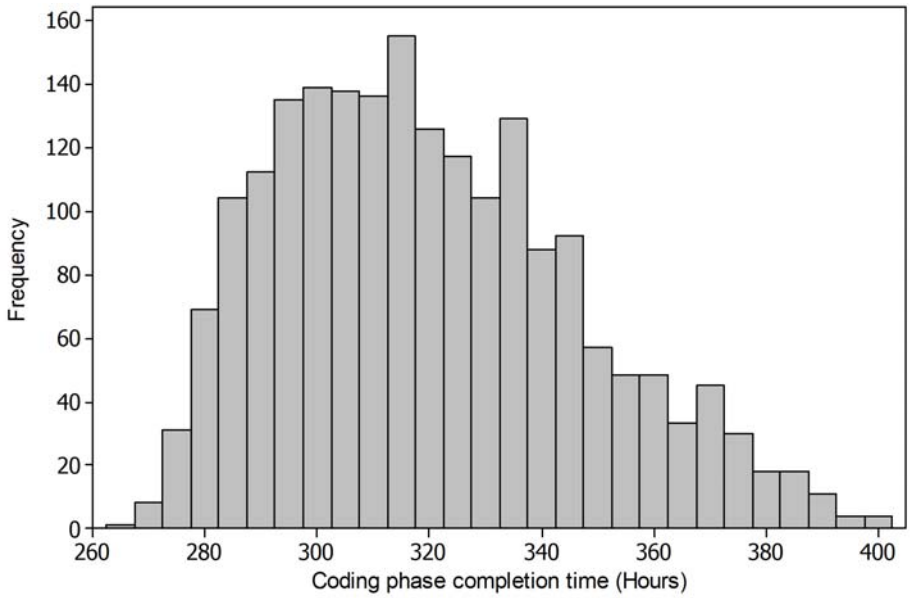


Figure 7: Coding phase completion time



The results from the simulations show that different learning curves generate variations in the time of completion of a project's coding phase. For the simulated scenarios, results show a completion time variation that ranges from 267 hours in the scenario with the shortest time, up to 402 hours for the longest, a difference of 135 hours between both ends of the spectrum. Additionally, the results display a positive asymmetric distribution at completion, as presented in the histogram of frequencies in Figure 8. Figure 8 is a horizontal cut when the SLOC pending to be developed reached cero, meaning the coding phase has been completed.

Figure 8: Distribution of coding phase completion times



7. Conclusions

In this article, a model was used to assess the implementation of frameworks in a software development project. Scenarios were simulated with different framework learning curves by using Monte Carlo simulations.

In the results, the range of values in the worst-case scenarios is greater than the range of values in the best-case scenarios. We consider this to be because the best-case scenarios quickly reach the learning curve asymptote, but they are also constrained by it. On the other end, the longest completion times are the result of scenarios where initial productivity is hindered because of the framework and a more difficult learning curve; in these cases, framework learning is low and there is also a slow progress in the project, increasing its completion time. The limit established by the framework learning curve asymptote is either not reached or reached until the end of the project.

Based on these findings, we believe a project manager must consider whether the application of framework will increase or affect developer productivity, and whether the framework learning curve will generate delays.

Different learning curves generate important variations in the time of completion of a project's coding phase. The results can serve as warning for projects unaware of the learning curve of a framework to be implemented. If the deadline for the project has limited flexibility, using a framework may delay the established completion date. Furthermore, the use of system dynamic simulation models makes it possible to observe the distribution of a project's

scenarios, which can be helpful in making better projections (of completions time, required effort, etc.) in projects using frameworks or other technologies.

Though the model and the results do not consider the measures the project manager can apply to rectify any deviation that may occur from the established plan, the model can be modified to include these policies and discover their effects. We estimate that the framework learning curve can intensify the negative effects of certain actions, for example, aligned to Brooks Law, if more people who are unfamiliar with the framework are added to make up for a delay in a project, the delay can be even greater.

8. References

Abdel-Hamid, T. K., & Madnick, S. E. (1991). *Software project dynamics: An integrated approach.* Englewood Cliffs, NJ: Prentice Hall.

Fayad, M., & Schmidt, D. C. (1997). Object-oriented application frameworks. *Communications of the ACM, 40*, 32-38. doi:10.1145/262793.262798

Kellner, M. I., Madachy, R. J., & Raffo, D. M. (1999). Software process simulation modeling: Why? what? how?. *Journal of Systems and Software, 46*, 91-105. doi:10.1016/S0164-1212(99)00003-5

Madachy, R. J. (2008). *Software process dynamics*. Hoboken, NJ: Wiley-IEEE Press.

Morisio, M., Romano, D., & Stamelos, I. (2002). Quality, productivity, and learning in framework-based development: An exploratory case study. *IEEE Transactions on Software Engineering, 28*, 876-888. doi:10.1109/TSE.2002.103322

Pfahl, D., & Lebsanft, K. (2000). Using simulation to analyse the impact of software requirement volatility on project performance. *Information and Software Technology, 42*, 1001-1008. doi:10.1016/S0950-5849(00)00152-X

Raffo, D. M., & Wakeland, W. (2008). *Moving up the CMMI capability and maturity levels using simulation* (Technical Report CMU/SEI-2008-TR-002). Retrieved April 8, 2013, from Carnegie Mellon University, Software Engineering Institute: http://www.sei.cmu.edu/library/abstracts/reports/08tr002.cfm

Sommerville, I. (2005). *Ingeniería de software* (7th ed). Madrid: Pearson Educación.

Sterman, J. D. (2000). *Business Dynamics: System Thinking and Modeling for a Complex World*. Boston: McGraw-Hill.

Wolstenholme, E. F. (2003). The use of system dynamics as a tool for intermediate level technology evaluation: Three case studies. *Journal of Engineering and Technology Management, 20*, 193-204. doi:10.1016/S0923-4748(03)00018-3