

Framework Based Software Development: Learning as an Investment Factor

Maurizio Morisio¹
University of Maryland
College Park – USA
morisio@cs.umd.edu

Ioannis Stamelos
Aristotle University
Thessaloniki – Greece
stamelos@csd.auth.gr

Daniele Romano
Politecnico di Torino
Torino - Italy

Corrado Moiso
CSELT
Torino - Italy

Abstract

We present a model for learning in framework based software development, based on a case study in framework based development. A framework is developed, then several applications are derived by customizing and extending the framework. We investigate the investments required, both in developing the framework and in allowing programmers to learn high level design and code to become proficient with it. The experiment shows an impressive variation in programmer's productivity before and after learning. An existing model is adapted to take learning investment into account.

Introduction

Software engineers have often experienced the sense of impotence due to developing an application that is (nearly) the same over several releases for different customers, without being able to capture and exploit the commonality.

Application frameworks are a possible solution to this problem. Based on object oriented technology, they are defined as “semi-complete applications that can be specialized to produce custom applications” (Johnson and Foote 1988).

Framework based development has two main processes: development of the framework, development of an application adapting the framework.

Frameworks are essentially a reuse technique. According to Karlsson's reuse model (Karlsson 1995), development of the framework corresponds to the FOR design activity, development of an application corresponds to the WITH design activity.

Earlier reuse models promoted the development of small grained reusable components (for instance procedures or classes) and their insertion in a reuse library. This model proved effective but limited. Applications are able to reuse small grained components only, reuse happens at the code level.

Frameworks are much larger (several related classes) and encapsulate high level designs. Reuse takes a complete different form, with much higher leverage. The application is built by slightly modifying the framework, and by accepting in toto the framework's high level design. In other words, most of the framework is reused, and reuse happens both at the high level design and at the code level.

The advantages promised by framework based software development are impressive, all explained by the high reuse levels expected: increases in productivity, shorter cycle time, increases in reliability. However, these advantages depend on several conditions to be met.

- The commitment by management to start and sustain an investment in framework development.
- The stability of the domain on a time range long enough to have a positive return on investment.
- The availability of highly skilled software designers and domain experts to build the framework.
- The availability of trained software engineers to develop the applications.

Therefore, deciding if building a framework is a complex managerial decision that has to be based on business and technical considerations. Economic models should guide such a decision process.

In this paper we analyse the learning issue, raised by the last point in the bulleted list above, and its impact on the economy of framework based development. It is common experience that a software engineer requires a considerable

¹ Maurizio Morisio is visiting researcher at University of Maryland, while on sabbatical leave from Politecnico di Torino.

amount of time to learn a framework so that he or she can use it effectively. The engineer has to become familiar with the domain, with the framework design, and with actual code. (We give it for granted that the software engineer is skilled in object oriented concepts and techniques. If this is not true, the learning effort is much higher, and so also the risks of failure).

The paper is organised as follows. We describe the context of an experiment in framework based development, the results and an economic model encompassing learning.

Context

Csel s.p.a. is a research and development company that works for Telecom Italia s.p.a., the main telecommunication services provider in Italy. Inside Csel, the network services division has developed in past years for Telecom Italia several advanced, value added prototypes of networking services (e.g. video on demand, multiconference, etc.). Based on experience gained with these prototypes, Telecom Italia decides whether and when to engineer and commercialize these services on national scale.

Csel observed the commonalities among the services developed in the past, and decided to experiment framework based development, both to gain know – how on the technique and to build estimation models (cost, return on investment, reliability, learning curve) to guide choices in the years to come.

In the following we describe the framework, and the services developed in the experiment.

The framework

The framework supports the development of multimedia services on a digital network. It uses a CORBA infrastructure, is developed in Java, and integrates COTS (Component Off-The-Shelf) such as CU-See-Me.

The framework is composed of a service platform, to abstract common functionalities from a network, and components to abstract control services.

The service platform offers these functionalities: service access (request to activate a service session, request to join a session), management of profiles of service subscribers, session control (coordinates the resources used in a session), network resource control. A Java API offers a common access to these functionalities offered by the service platform. Other APIs offer access to special resources (ex. reflector, video bridge, vocal gateway) that are encapsulated as Corba IDLs. This design allows to add later special resources to the framework, if needed.

The framework is designed to be reused black-box (Fayad and Schmidt 1997), except for GSS's that have to be specialised from the base class (white-box). The framework is composed of 22 classes, or around 10KSLoc (source lines of code).

The services developed

During the experiment the following applications (or services) were developed. Sometimes a service differs only slightly from another. This is called a variant and is treated separately in the experiment.

SPY – Spy Camera – This service allows to monitor a number of video cameras on different sites.

SPY Multi – Spy camera multi observer. Variant of Spy camera. Same as Spy camera, but several observers can be on a session.

TLL – Telelearning – This service allows to set up a distant learning session.

AUC – Auction – This service offers an Internet based auction.

AUC Browser – Auction with browser – Variant of Auction. The only difference is that when the first object is auctioned, an internet browser is used.

MCS – Multiconference – Audio and video multiconference on the Internet.

MCS Mail – Variant of Multiconference. In MCS, if the chairman calls a participant and can't find it, nothing more happens. In this variant, an email is sent to the participant to notify him.

MCS Sms – Variant of Multiconference. An SMS message is sent when the participant is not found by the chairman.

VOD – Video on demand. Selection, payment and playing of a movie.

Results

Figure 1 contains a summary of experiment factors and values of output variables. Services appear in the temporal order in which they were produced. The two top rows refer to services developed without framework, the third row refers to framework development, the rest refers to services developed with the framework. For the latter, size counts (Classes, methods, Locs) refer only to code developed in addition to the framework, which is always reused verbatim, in toto.

Service	Framework	Variant	Programmer	Language	Effort (hours)	Classes	Methods	LOC	Productivity (Loc/hour)
MCS	N	N	A,B,C	C++	400			15430	38.58
VOD	N	N	A,B,C	C++	240			11350	47.29
<i>FWK</i>	<i>NA</i>	<i>NA</i>	<i>A,B</i>	<i>Java</i>	<i>2720</i>	<i>22</i>	<i>218</i>	<i>9818</i>	<i>3.61</i>
SPY	Y	N	D	Java	95	17	76	12331	129.80
TLL	Y	N	D	Java	87	17	97	13286	152.71
SPY Multi	Y	Y	D	Java	18	18	80	12974	720.78
MCS	Y	N	D	Java	63	12	74	12439	276.42
AUC	Y	N	D	Java	45	19	96	13633	216.40
AUC Brow	Y	Y	D	Java	13	20	103	14042	1080.15
MCS email	Y	Y	D	Java	16	14	82	12707	794.19
MCS sms	Y	Y	D	Java	16	14	80	12865	804.06
VOD	Y	N	D	Java	34	17	76	12231	359.74

Figure 1. Values of factors and variables

The productivity for variants is much higher than for services. We will treat them separately, as they are different development approaches. If we plot the productivity figure for services (Figure 2), we remark a constant and remarkable increase. All factors are constant except learning. Moreover, SPY and VOD are two services with very similar functionality and structure and can be considered virtually equivalent. They were developed as first and last to simulate the replication of one development. Considering SPY and VOD, the productivity increase is an astonishing 280%. We interviewed programmer D to cross check our interpretation of data. He confirmed that improved knowledge of the framework was to be considered the major responsible of the change in productivity. He was aware of having improved his knowledge of Java too, but he considered its effect as marginal. In his opinion the learning effect was over after developing VOD.

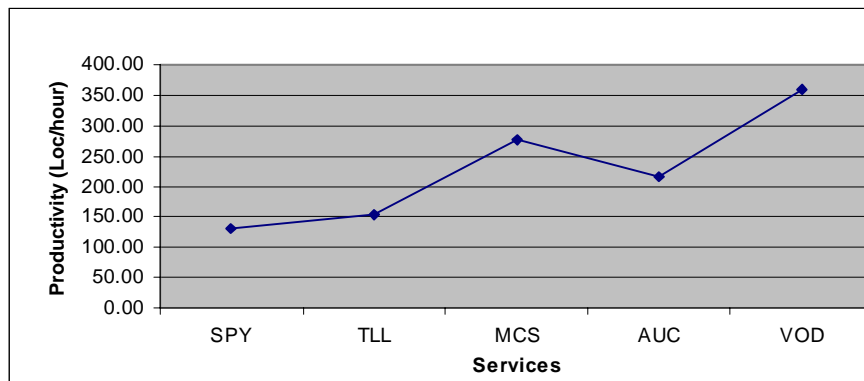


Figure 2. Productivity for services development.

Another interesting result is the productivity for developing the framework (3.6 Loc/hour), as compared with the productivity of non framework based development (38.7, 47.2), a ratio of more than 10 times. Given that the languages used belong to the same class, and the programmers are the same, we argue that the reason is the inherent complexity and difficulty of developing a framework. It should be noted, however, that programmers A and B were developing their first framework.

An economic model

A model suitable for our purposes is described in (Poulin 1997) and was introduced by the U.S. Defense Information System Agency (DISA). According to this model the total effort expressed in labor hours in order to develop a project is given by the formula:

$$E_r = a_1 L_n + a_2 L_r + a_3 L_v + a_4 L_a \quad (1)$$

where:

L_n = lines of new custom code

L_r = lines of new reusable code

L_v = lines of verbatim reused code

L_a = lines of adapted reused code

The coefficients represent the relative costs of reuse in terms of development hours per line of code as follows:

a_1 = relative cost of writing custom code.

a_2 = relative cost of writing reusable code.

a_3 = relative cost of verbatim reuse.

a_4 = relative cost of adapted reuse.

In (Poulin 1997) the following default values are proposed for these coefficients:

$a_1 = 1.0$, $a_2 = 1.5$, $a_3 = 0.2$ and $a_4 = 0.5$. It is stated that they can vary significantly according to the development environment.

This model assumes that no learning effect is present: it holds for projects developed by experienced personnel.

For the applications developed with the help of the framework, the framework was totally verbatim reused and there was no adapted reused code. The programmer did not write reusable code. The new code was developed in order to make use of some methods provided by the classes of the framework. The influence of the learning factor can be determined if we calculate the way the coefficient a_1 changes across the projects. For our purpose the model described by equation 1 is adapted. The total effort in labor hours to develop an application using this framework is given by:

$$E_r = a_1 L_n + a_3 L_v \quad (2)$$

where:

L_n = Lines of new code developed in order to make use of the methods provided by the framework,

L_v = Lines of code of the framework (verbatim reused),

and:

a_1 = the relative cost of writing code that implements calls to the functions provided by the framework, in development hours per line of code,

a_3 = the relative cost of verbatim reused code in development hours per line of code.

It is important to note that learning has no effect on the cost of verbatim reused code, since it is a repetitive task accomplished in a standard way. The value of the coefficient a_3 can be considered to be constant and approximated as follows.

From the default values of formula 1, we note that the ratio a_1/a_3 is equal to 5: the cost of writing new code is five times higher than the cost of verbatim reusing code. However, in our case, this may be true only for the last application (VOD), since, according to the programmer, this application signals the end of the learning effect.

Therefore the relative cost of verbatim reuse may be approximated by the formula 2 considering $a_1/a_3=5$:

$$a_3 = \frac{E_{VOD}}{L_{v,VOD} + 5L_{n,VOD}} \quad (3)$$

resulting in $a_3=0.0018$, after replacement of values. We will use this value for all five projects.

The relative cost of writing custom code (i.e. code which uses the framework) is determined by the formula:

$$a_1 = \frac{E_r - a_3 L_v}{L_n} \quad (4)$$

Using equation 4 we calculated the relative cost of writing code that makes use of the framework for the five applications. The results are presented in Figure 3.

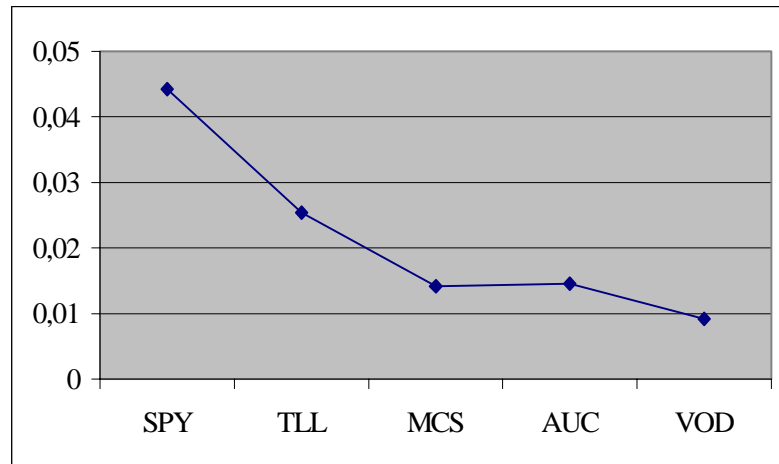


Figure 3. The relative cost of writing new code

From this diagram it is evident that the cost of writing new code is higher for the first two projects, during which the programmer learns how to use the framework, and decreases as the number of the applications developed using the framework increases. The diagram fits well the Wright curve, the earliest industrial learning curve.

Conclusion

We have presented an experiment in framework based development. The study produced these main findings.

- A ratio of 280% between productivity before and after fully learning the framework.
 - A more than tenfold ratio between productivity for normal development and framework development
- These findings are peculiar to a specific context, with a single developer, and cannot be generalized to other contexts.

An economic model to support management decisions and including the effect of learning has been proposed. Future work will be dedicated to developing a comprehensive model for return on investment analysis, considering learning, effort and reuse level.

References

- Box G.E., Hunter J.S., Hunter W.G., *Statistics for Experimenters: an Introduction to Design, Data Analysis and Model Building*, John Wiley, 1978.
- Fayad M.E., and Schmidt D.C., Object Oriented Application Frameworks, *Communications of the ACM*, 40(10), October 1997, 32-38.
- Johnson R., and Foote E., Designing Reusable Classes, *Journal of Object Oriented Programming*, (1)5, June 1988.
- Karlsson, E.A., *Software Reuse*, John Wiley, 1995.
- Poulin, J.S. "Measuring Software Reuse: Principles, Practices and Economic Models", Addison Wesley 1997.
- Taguchi G.D., Yokoyama Y, *Taguchi Methods: Design of Experiments*, 1993.