

Introducing Legacy System Migration Technologies in an Academic Context: a Controlled Experiment

Massimo Colosimo, Andrea De Lucia, and
Genoveffa Tortora
*Dipartimento di Matematica e Informatica,
University of Salerno, Via Ponte Don Melillo,
84084, Fisciano (SA), ITALY*
cillo_81@yahoo.it, {adelucia, tortora}@unisa.it

Giuseppe Scanniello
*Dipartimento di Matematica e Informatica,
University of Basilicata, Viale Dell'Ateneo,
Macchia Romana, 85100, Potenza, ITALY*
giuseppe.scanniello@unibas.it

Abstract

To verify whether or not migration technologies can be adopted, systematic and quantitative evaluations should be performed. In this paper we present the results of a controlled experiment aimed at assessing the usefulness of an Eclipse plug-in, named MELIS (Migration Environment for Legacy Information Systems). This plug-in has been developed to support the migration of legacy information systems to a web-enabled multi-tier target architecture according with an incremental migration strategy. The context of the experiment was constituted of master students in Computer Science at the University of Salerno. The subjects without COBOL programming experience, while half of them had J2EE programming experience. The results of the experiment confirmed that the use of MELIS increases the productivity with respect to the use of traditional development tools.

1. Introduction

Research in the empirical software engineering field aims at acquiring general knowledge about which process, method, technique, language, or tool is useful for whom to conduct which tasks in which environments [5][18][19][31]. In particular, empirical studies should be devised to transfer software engineering technologies produced in research laboratories to practitioners and to evaluate whether they potentially fulfill the industry needs [6][22][28].

To verify whether or not software engineering technologies can be adopted, systematic and quantitative evaluations should be performed in terms of surveys, controlled experiments, and case studies [6][39]. Among these evaluation techniques controlled experiments are recognized to be important means since they represents the classical scientific method for identifying cause-effect relationships [31].

In [16] we presented the results of a research project aimed at defining methods and at developing tools to migrate the legacy systems of a small/medium Italian software enterprise towards modern web architectures. In particular we defined a migration strategy and developed an Eclipse plug-in, named MELIS (Migration Environment for Legacy Information Systems), to support the migration of the graphical user interface, and in the restructuring and wrapping of the original legacy code. The plug-in and the migration strategy have been evaluated in a case study [16] conducted on one of the more relevant legacy system of the software enterprise. The management of the software enterprise also required to experiment the usefulness of the migration technology. To this end two controlled experiments were concurrently carried out. In particular, the contexts of these experiments were constituted of master students and professional programmers of our industrial partner, respectively. We varied the context variables in the environments in which the outcomes were achieved to identify potentially important environmental factors that could affect the use of MELIS, thus understanding and improving its external validity [7].

In this paper, we present the controlled experiments performed with master students. It was mainly due to number of available observations that are larger than the number of observations concerning the controlled experiment carried out in with professional programmers. The data analysis of the presented controlled experiment revealed that the productivity is increased of a factor 5.6. Moreover, according with the outcomes of the other controlled experiment¹, we also observed that the use of traditional development tools require more time to perform the migration tasks when the subjects do not have specific competences.

The remainder of the paper is organized as follows: Section 2 presents MELIS and the migration strategy. Section 3 illustrates the controlled experiment and discusses the results, while related work is discussed in Section 4. Final remarks and future work conclude the paper.

2. Related Work

A significant number of researches have been carried out on the definition and development of tools for the migration of legacy systems [10][11][12][13][29][33][35], and recently on the migration to web technologies [3][4][9][14][26][36]. A classification of the current approaches to evolve LISs toward new technological infrastructures was described in [29]. The authors also suggested guidelines and a methodology for the migration of transaction oriented systems together with some tools to reduce the time, the human resources, the global costs, and risks of a migration process. Both the tools and methodology have been assessed in a set of laboratory case studies.

Nowadays the migration of legacy systems to the web is widely investigated [3][4][8][9][36]. Many of the strategies proposed are referred to decomposable or semi-decomposable software systems. In particular, Aversano *et al.* [3][4] proposed the main results of a migration project aimed at integrating an existing COBOL system into a web-enabled infrastructure. The original system was semi decomposable with a client component (represented by

¹ www.scienzemfn.unisa.it/scanniello/CExp_1/index.htm

the user-interface) and a server component (represented by the application logic and database). The Graphical User Interface was manually migrated to Microsoft ASP, while the server component was wrapped with dynamic load libraries.

Bodhuin *et al.* in [8] described an approach to migrate a COBOL system easy to decompose towards a web-enabled architecture based on Model View Controller (MVC). The software components of the original LIS are identified using slicing techniques. These components are restructured and then turned into JAVA classes by using the PERCobol tool [21]. The same authors have presented in [9] an approach and a tool also based on PERCobol to migrate a non-decomposable LIS to two-tier web-enabled architecture. A Screen Proxy was introduced for the management of the requests coming from or going to the user interface.

Generally, the migration of LISs to the web involves two main activities: wrapping of the original legacy code and reengineering of user interfaces. The wrapping techniques have a relevant role in the migration [2][3][4][8][9][32] and in the decomposition into client and server components [12][13] of LISs as well. For example, wrapper interfaces have been used by legacy or ex-novo software components to use the encapsulated components in a transparent way. Sneed [35] proposed tools and wrapping techniques, where online programs were transformed into data driven subprograms which process an XML-document. Differently, batch programs were adapted to read and write XML-documents. Finally, in subprograms parameters were set from an XML-document. Recently, Sneed in [37] has been extended this approach to enable the integration of a legacy system into a Service Oriented Architecture.

Regarding the reengineering of user interfaces, in the literature interesting researches based on data-flow analysis [24] and State Transition Diagrams [10][38] as well as on techniques derived from artificial intelligence [25] have been proposed. For instance, Moore and Moshkina [26] described an architectural restructuring aimed at migrating character-oriented user interfaces to a web based front end. The reengineered interfaces were event driven to better respond to the users' interactions. To meet the growing needs of a business company, a technique for reengineering user interfaces as a consequence of the LIS wrapping is proposed by Sneed in [33]. An approach to migrate character based user interface to any client devices is proposed in [10]. This approach exploits a black-box technique for capturing the dynamic and static models of the user interfaces and reproduces them on client devices with the support of a software wrapper. XML and XSL technologies are used to reproduce the LIS original user interface. Also in this case a wrapper was designed to satisfy service stability, data integration, and application integration requirements. In this paper the authors also presented both a methodology and a toolkit to design the wrapper and to support its application, respectively. The approach took into account user interfaces based on characters and forms, whereas the checks are performed when all the input fields are filled in. An extension of the approach to a Service Oriented Architecture has been proposed in [14].

In most cases, the migration methods and tools presented in the literature have been assessed in case studies. The drawback of this empirical method is that determining trends and statistical validity is often difficult because each development is relatively unique. As a consequence, it is difficult to compare two different development profiles

[40]. To the best of our knowledge no controlled experiment has been published concerning the effectiveness and usefulness of software migration technologies.

3. Migration Process and Tool

Due to the assessment of the legacy systems of the software enterprise, which revealed that these systems have a very low decomposability degree [11], we adopted a migration strategy to reengineer the user interface using web technology, turning interactive legacy programs into batch programs and encapsulating them using a communication wrapping layer. The wrapper is in charge of enabling the new user interface to interoperate with the restructured and wrapped legacy program. The adopted solution enables an incremental migration of the legacy programs or subsystems composing the LIS, thus preserving the past investments and reducing risks and costs of a mass replacement [11][34].

Figure 1 shows the adopted target architecture. The *Wrapper* component is a middleware enabling the communication between the *Restructured LIS* and the reengineered user interface. The original LIS is turned into a batch program (Restructured LIS) and all the interactions with the user interface are redirected to the wrapper component. The legacy user interface is reengineered in two components, namely the *Reengineered GUI* and the *GUI Deliverer*. The Reengineered GUI includes the JSP pages replacing the original user interfaces of the LIS, while the GUI Deliverer includes Java Servlets and Beans used to manage the control logic of the web user interface and access the functionalities of the Restructured LIS. The GUI Deliverer accesses the Reengineered GUI to get the web pages required to accomplish a given functionality. The communication with the wrapper is implemented using either RMI (Remote Method Invocation) or SOAP (Simple Object Access Protocol).

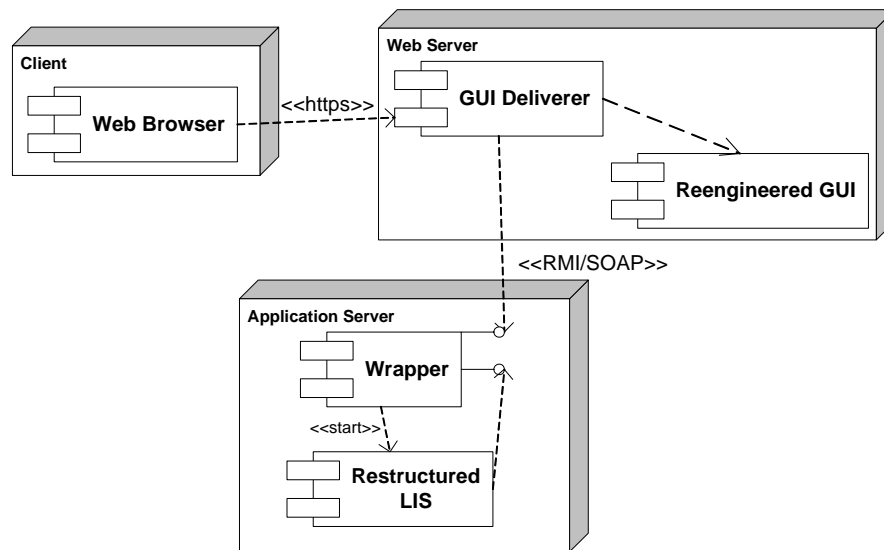


Figure 1. The target software architecture

In order to use the Wrapper in the migration of any LISs, we developed it as a generic component. In particular, it is implemented as a Dynamic Load Library (DLL) and provides two different interfaces to the GUI Deliverer and to the Restructured LIS, respectively. The wrapper is also in charge of starting the Restructured LIS, when a user accesses the first time the system and manages the synchronization of the execution between the new web based user interface and the Restructured LIS. More details on the target architecture can be found in [16].

Figure 2 shows the migration process in terms of an activity diagram with object flow. The Pre-processing phase is semi-automatically supported by MELIS and provides details on the types and attributes of each graphical object composing a SCREEN SECTION (a specific COBOL section used to define dialogs with users). The structure of the SCREEN SECTIONS and the corresponding graphical objects are encoded in a XML file. In particular, this file hierarchically organizes SCREEN SECTIONS, graphical objects, and the control checks of a given COBOL program. The first level of the hierarchy reports the names of the SCREEN SECTIONS, while their graphical objects and the corresponding BEFORE and AFTER clauses are shown at second and third level, respectively. These clauses are represented as leaf nodes of the hierarchy together with the names of the associated procedures implementing the corresponding field checks.

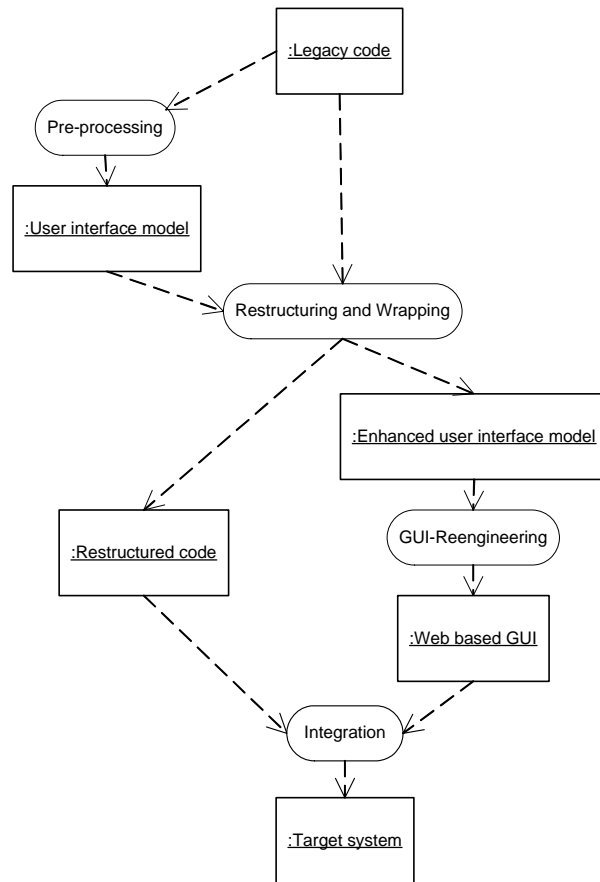


Figure 2. The migration process

The XML file is automatically produced by MELIS and is successively enhanced by the developers according to the identification of the checks that can be migrated to the client. To this end the BEFORE and AFTER clauses are classified as: format check operations, which are migrated to the client, and field database access or application logic operations, which are left on the restructured COBOL program and invoked using the Wrapper component. The classification of the BEFORE and AFTER clauses is suggested by MELIS to provide developers a semiautomatic support. To this end, MELIS highlights the original legacy code in yellow, when the checks could be migrated to the client side, in red otherwise. The format checks are embedded in the XML file in terms of javascript functions. To further support the developer, MELIS also proposes a library of reusable and customizable javascript functions, which can be used in the migration of format check operations.

During the Restructuring and Wrapping phase the LIS code is restructured to become a batch program. This is the most challenging phase of the migration process and is semi-automatically supported by MELIS. In particular, it automatically comments (through a “*”) the statements DISPLAY (used to visualize a SCREEN SECTION) and ACCEPT (used to get data from a SCREEN SECTION) and then replaces them by calls to the Wrapper component. Transitions between SCREEN SECTIONS are preserved in the reengineered graphical user interfaces as well as the error handling. An ACUCOBOL-GT compiler has also been integrated within MELIS to re-compile the restructured ACUCOBOL-GT program.

The *GUI-Reengineering* phase produces the web based graphical user interface starting from the intermediate SCREEN SECTION representation. In particular, the software engineer uses MELIS to generate a JSP page for each SCREEN SECTION of a given ACUCOBOL-GT subsystem. Moreover, to enable the communication between each JSP page and the restructured legacy code both a servlet and a Java bean are also generated. The Java bean contains the fields of the SCREEN SECTIONS and is built from the XML description contained in the enhanced user interface model.

It is worth pointing out that the JSP pages generated by MELIS are very similar to the original one, but they can be eventually modified or enhanced by using the MELIS authoring feature. To obtain the more appropriate look and feel and to easily modify the reengineered user interface once the restructured LIS has been integrated and deployed Cascading Style Sheets (CSS) has been adopted in the reengineered graphical user interfaces.

Finally, the *Integration* phase enables the integration of the restructured legacy code and reengineered web based graphical user interfaces. In this phase the developer first compiles the restructured legacy code using the ACUCOBOL-GT compiler and then deploys the reengineered web based graphical user interfaces on the *Web Server* node and the compiled code of restructured legacy program on the *Application Server* node. MELIS fully supports the compilation of the restructured legacy code and the deployment of the reengineered user interface.

4. The Controlled Experiment

In this section we describe the controlled experiment following the template suggested by Wohlin *et al.* [39].

4.1 Experiment definition and context

In this work we analyze and evaluate the usefulness of MELIS in the migration of legacy systems towards a web based multi-tier architecture and the possibility of adopting MELIS by programmers without a deep knowledge of the COBOL programming language.

The subjects of the experiment were twenty eight master students in Computer Science at the University of Salerno. The subjects were volunteers with comparable background except for the knowledge on the J2EE technology. Indeed, fourteen had J2EE programming experiences, while the remaining subjects have not used this technology before the experiment. Since the two groups of students had homogenous background, we randomly grouped them in fourteen development teams each composed of a subject familiar with J2EE technology and a subject not familiar with this technology. All the recruited subjects were not familiar with MELIS as well as the supported migration strategy.

The experiment has been performed online at the University of Salerno in the software engineering laboratory. The PCs used in the experiment had a comparable hardware configurations and the same operating system. Due to the number of subjects and available PCs, we performed the controlled experiment in two days. In particular, sixteen subjects carried out the controlled experiment on the 20th of October 2006, while the remaining the 23rd of October 2006.

The following are the selected migration tasks that the development teams had to carry out during the controlled experiment:

- **T₁**: migrating an ACUCOBOL-GT program computing the basic arithmetic operations, i.e. addition, subtraction, multiplication and division; (196 LOCs)
- **T₂**: migrating an ACUCOBOL-GT program to achieve the age of a person considering his/her birthday. This program presents three embedded controls to check the inserted month, day, and year. (238 LOCs)

The tasks T₁ and T₂ were constituted of 196 and 238 LOCs, respectively and were selected to be accomplished within 3 hours and a half and in order to propose the majority of the problems that a developer could encounter in the migration of an ACUCOBOL-GT program. Once the development teams accomplished the tasks, the restructured legacy code and the reengineered graphical user interfaces have to be integrated and deployed within the target architecture.

Two different migration methods, i.e. *PL* and *NOPL*, are used to perform the migration tasks. Both methods migrate the legacy programs to the target architecture shown in Figure 1 that includes the built-in wrapper component, according to the migration process in Figure 2. The difference is that *PL* is performed using the MELIS Eclipse plug-in, while *NOPL* is performed using the development environments: ACUBENCH [1], an ACUCOBOL-GT environment (used to restructure and wrap the original legacy code), and Lomboz [23], an Eclipse plug-in to develop J2EE applications (used for the reengineering of the web based user interface).

4.2 Hypothesis

This controlled experiment aims at confirming that on average the use of MELIS improves the developers' productivity in the migration of ACUCOBOL-GT programs with respect to the use of traditional development tools. Therefore, we have formulated the following null hypothesis:

- **H_{n1}**: the use of MELIS does not significantly affect the effort to migrate an ACUCOBOL-GT program to the web;

The alternative hypothesis is:

- **H_{a1}**: the use of MELIS significantly affects the effort to migrate an ACUCOBOL-GT program to the web;

The migration effort includes both the effort to comprehend the original ACUCOBOL-GT program and the times required to carry out all the phases of the migration strategy.

4.3 Selected variables and experiment design

In order to properly design the experiment and analyze the results, we considered the following independent variables:

- **Method**: this variable indicates the factor on which the study is focused, i.e. PL and NOPL.
- **Task**: the migration tasks described in Section 4.1.
- **Lab**: the experiment is organized in two laboratory sessions, i.e. Lab1 and Lab2.

The dependent variable is the time required to migrate the assigned tasks, as mentioned before it includes the times to comprehend the migration task and to carry out the phases of the migration process.

All combinations of the factors Method (PL and NOPL) and Task (T₁ and T₂) represent the considered treatments. To avoid results to be biased by group ability, each group experienced both Methods and both Tasks over two subsequent laboratory sessions (*Lab1* and *Lab2*). Also, to minimize the learning effect, we needed to have groups starting to work in Lab1 with or without MELIS on both the migration tasks. Table 2 summarizes the experiment design.

4.4 Preparation

As mentioned above, the number of development teams was much more than the number of experiment supervisors and available PCs, thus we divided the subjects in two groups and carried out the controlled experiment in two days. Details on the preparation of the controlled experiment in terms of activities and their schedule are reported in Table 1. In particular, the first and the second rows of this table contain the activities of the first and second group, and the corresponding laboratory sessions, respectively. The subjects of the two groups together attended a training session on the migration strategy and MELIS, and then on the ACUCOBOL-GT programming

language. Differently, the first group carried out the controlled experiment the 20th of October 2006, while the second group the 23rd of October 2006.

The two groups of subjects also attended a training session separately on the controlled experiment before accomplishing both the laboratory sessions. The training session aimed at providing all the recruited subjects an equal prior knowledge and to deeply describe the software applications, target architecture, and migration process. To let subjects more confidence with the MELIS plug-in some examples (not related to the tasks to avoid biasing the experiment) were also presented. The training sessions were concluded presenting detailed instructions on the tasks to be performed.

Table 1. Experiment schedule

		10/17/2006	10/18/2006	10/20/2006		10/23/2006	
		Morning	Morning	Morning	Afternoon	Morning	Afternoon
First group	Introduction of the migration strategy and of the MELIS plug-in	Introduction of ACUCOBOL-GT	Training session	Lab1	Lab2		
Second group	Introduction of the migration strategy and of the MELIS plug-in	Introduction of ACUCOBOL-GT				Training session	Lab1 Lab2

Table 2. Experiment design

		Groups			
		A	B	C	D
Lab1		T1_PL	T1_NOPL	T2_PL	T2_NOPL
Lab2		T2_NOPL	T2_PL	T1_NOPL	T1_PL

In order to assess the usefulness of MELIS, its ease of use, and how much time subjects spent using it, at the end of each laboratory session, the survey questionnaire shown in Table 3 has been presented. The questionnaire is composed of questions expecting closed answers according to the five-point Likert scale [27]: from 1 (strongly agree) to 5 (strongly disagree).

Regarding the preparation of the PCs of the software engineering laboratory used in the controlled experiment, we installed MELIS, ACUBENCH [1], and Lomboz [23]. ACUBENCH and Lomboz were used for the NOPL method.

4.5 Material and execution

The development teams accomplished each laboratory session without time limit. Once the tasks of the laboratory sessions were migrated the teams filled in the survey questionnaire, while the supervisors collected the

software artifacts produced during the laboratory sessions, namely the restructured ACUCOBOL-GT programs and the reengineered graphical user interface. The supervisors also collected the log files containing the information traced by MELIS.

Table 3. Survey questionnaire

Id	Question
Q1	I had enough time to perform the assigned task
Q2	The objectives of the assigned task were perfectly clear to me
Q3	The task I had to perform was perfectly clear to me
Q4	The material given to me provided enough information concerning the task to perform
Q5	I believe that the restructuring of the original ACUCOBOL-GT program was simple
Q6	I believe that that the reengineering of the original graphical user interface was simple
Q7	I believe that the integration and the deployment of the migrated ACUCOBOL-GT program was simple
Q8	How much time (in terms of percentage) did you spend to restructure the original legacy code (A < 25% - B >= 25% and < 50% - C >= 50% and < 75% - D >= 75%)?
Q9	How much time (in terms of percentage) did you spend to reengineer the user interface (A < 25% - B >= 25% and < 50% - C >= 50% and < 75% - D >= 75%)?
Q10	How much time (in terms of percentage) did you spend to integrate and to deploy the migrated ACUCOBOL-GT program (A < 25% - B >= 25% and < 50% - C >= 50% and < 75% - D >= 75%)?

In order to carry out the controlled experiment each development team was provided with a folder containing a pencil, some white sheets, and the following hard copy material:

1. The introductory presentation of the experiment;
2. Depending on the method, we provided the MELIS user manual and the guideline to migrate the assigned task or a set of user guidelines to migrate ACUCOBOL-GT programs without the MELIS support;
3. The source code of the ACUCOBOL-GT programs, to be migrated in the two tasks;
4. The survey questionnaires to be filled in at the end of the two laboratory sessions.

4.6 Experiment results

The raw times collected at the end of the controlled experiment are shown in Table 5. This table details the times expressed in minutes, grouped by laboratory sessions, treatments, and development teams. Instead, the descriptive statistics of the experiment are reported in Table 4. As shown in this table, the development teams spent on the

average about 47 minutes to migrate the assigned ACUCOBOL-GT programs using MELIS, while they spent on average about 262 minutes without using it. Therefore, the time to migrate a program using the MELIS plug-in is on the average 17.86% of the time to migrate the same program without the plug-in. Moreover, let us also note that the time to migrate the task T_2 is more than the time to migrate the task T_1 for both the treatments.

Table 4. Descriptive statistics

Task	Method	Min	Max	Mean	Median	Std. Dev.
All	NOPL	174.00	335.00	262.3571	296.5	52.59220
	PL	25.00	72.00	46.8571	48	14.44809
T1	NOPL	174.00	335.00	250.5714	227	58.96569
	PL	25.00	64.00	45.4286	49	15.37159
T2	NOPL	192.00	314.00	274.1429	290	46.81677
	PL	34.00	72.00	48.2857	47	14.53403

The times to perform the migration tasks according to the considered treatments are summarized by the boxplots in Figure 3. This figure does not present outliers and shows a good distribution of the time to accomplish the tasks T_1 and T_2 using MELIS. Also, the boxplots of the NOPL treatments do not present outliers. The boxplots of the treatments NOPL are more skewed than the boxplots of the treatments PL, while the task T_1 has a distribution more symmetric than the distribution of the task T_2 . Thus, we further tested the normality of the raw times according to the considered methods PL and NOPL using both Kolmogorov and Shapiro-Wilk [30] tests. These tests revealed that the null hypothesis (i.e. the sample is taken from a normal distribution) cannot be rejected, since the p-values are larger than 0.05.

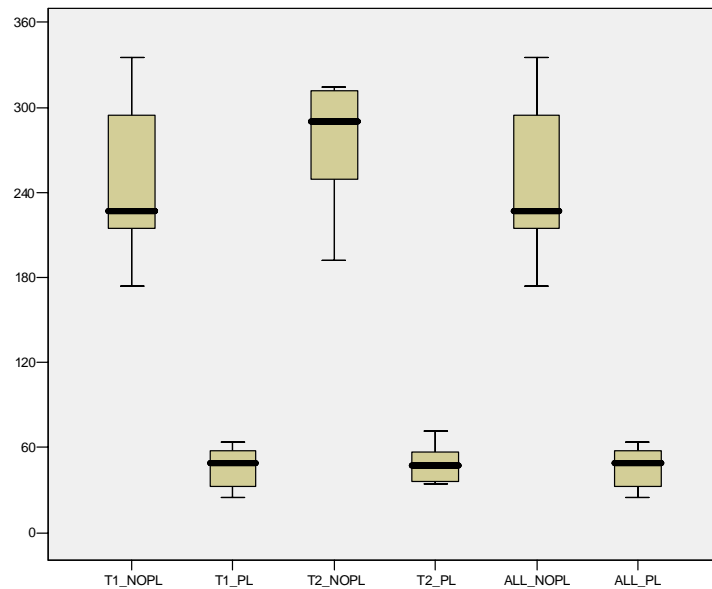


Figure 3. Times to perform the migration tasks

Concerning the hypotheses of the controlled experiment, the H_{n1} null hypothesis can be rejected with a significance level of 95% as the Wilcoxon paired test [15] revealed. In particular, the results of this test indicate that there is a statistically significant difference between the use or not of the MELIS plug-in on the time to migrate the assigned tasks ($z = -4.623$, $p = 0.000$).

We also used a two-way Analysis of Variance (ANOVA) [17] to investigate the effect of the selected dependent variable and the considered factors. We adopted this technique to test null hypotheses about the effects of other variables on the means of various groupings of a single dependent variable. We also investigated the interactions between the factors as well as the effects of individual factors. In particular, the results of the two-way ANOVA test (see Table 6) revealed that the interaction between Method and Task is not significant. On the other hand, there is very highly significant (p -value = 0.000) effect of the Method overall, while the Task factor is not significant (p -value = 0.380). Further ANOVA analyses revealed no significant dependency of the similarity on the Lab (indicating absence of learning effect) and on the considered teams. We also applied the Friedman non-parametric test to confirm the results of the two-way ANOVA test. In particular, this test revealed that the Method variable and the dependent variable are related (p -value = 0.000). Moreover, this test also indicates that there is a statistically significant relation of the distributions of the dependent variable and the Task variable. This could be essentially due to the embedded control flow of the task T2.

Table 5. Raw data expressed in minutes

	Groups													
	A				B			C				D		
	Team1	Team2	Team3	Team4	Team1	Team2	Team3	Team1	Team2	Team3	Team4	Team1	Team2	Team3
Lab1	59	30	64	49	212	227	335	35	37	72	47	234	192	314
Lab2	290	313	265	311	51	34	62	217	174	274	315	35	25	56

Table 6. ANOVA table of similarity by Method and Task ($R^2 = 89,9\%$, $R^2(\text{adj}) = 88,7\%$)

Source	Type III Sum of Squares	Df	Mean Square	F	p-value
Method	325081.75	1	325081.75	212.601	0.000
Task	1222.321	1	1222.321	0.799	0.380
Interaction	750.893	1	750.893	0.491	0.490
Residual	36697.714	24	1529.071		
Total	1033047	28			

The data collected from the survey questionnaire are summarized by the boxplots of Figure 4. In particular, the answers of the survey questionnaire corresponding to the tasks T_1 and T_2 are summarized by the boxplots on the left hand side and on the right hand side, respectively. The analyses of the collected data showed that the time to

perform the tasks T_1 and T_2 was considered appropriate as well as their objectives (see boxplots of the questions Q1 and Q2). The subjects also considered clear the assigned tasks as the answers of the question Q3 revealed. A positive judgment was also expressed on the hard copy material provided as support to accomplish the tasks T_1 and T_2 .

The boxplots of the question Q5 revealed that the subjects manifested different judgments on the program restructuring simplicity. In fact, the answer distributions are skewed as the tails of these boxplots shown. Regarding the question to assess the simplicity to migrate the graphical user interface of the ACUCOBOL-GT programs (question Q6) the subjects expressed a good judgment on the average.

The agreement level regarding the integration and the deployment of the migrated programs of the tasks T_1 and T_2 was quite concordant and adequate as the boxplots of the question Q7 revealed. However, results showed (with p-value 0.004 computed with Mann Whitney U-test) that subjects performing the task T_1 felt simpler the integration and the deployment of the migrated program than the subjects performing the task T_2 . Once again this could be due to the embedded control flow of the task T_2 . We also used the Mann Whitney U-test to verify that the subjects found simpler the integration and deployment phase when the MELIS plug-in was adopted (p-value 0.022) to migrate the assigned tasks. The answer analysis of Q10 revealed a good agreement level on the integration and deployment phase. It is worth mentioning that considerations on the question Q8 and Q9 cannot be performed because of the distribution of the corresponding answers.

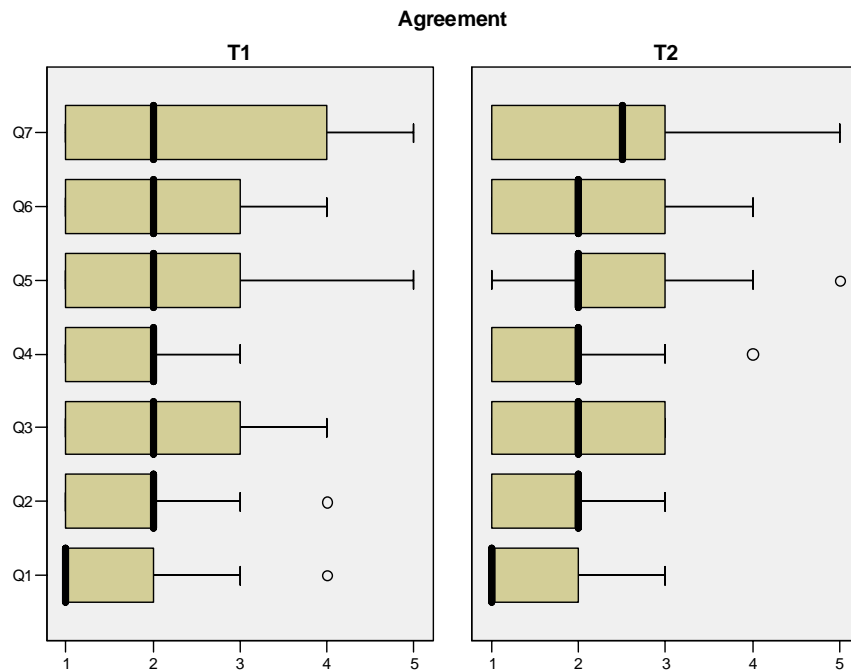


Figure 4. Boxplots of the survey questionnaire

4.7 Threats to validity

In this section we describe the threats to validity (i.e. internal, construct, external, and conclusion) that could affect the results of the controlled experiment.

The internal validity threats are relevant for our study since we aimed at concluding that MELIS made a difference in the migration of LIS to the web according to the proposed migration strategy. The key question in internal validity was whether observed changes can be attributed to the learning effect and not to other possible causes. The *internal validity* threats are mitigated by the experiment design, since each group worked, over the two laboratory sessions, on two different migration tasks and with two different Methods (PL and NOPL). This result was confirmed by the two-way ANOVA test, which did not reveal a significant learning effect of the subjects across the laboratory sessions. On the other hand, the survey revealed that the subjects found clear everything regarding the migration tasks. Although the subjects applied the migration strategy and adopted the target architecture, they know neither the objective of the experiment nor its hypothesis. Finally, we did not evaluate subjects on their performance within the laboratory sessions at the end of the experiment.

The proper design of the experiment also enabled to mitigate the *construct validity* threats. Depending on the treatment, the measurement of the dependent variable was performed either analyzing the log files produced by MELIS or considering the times gathered by the supervisors. Finally, the survey questionnaire was designed using standard ways and scales [26].

External validity refers to the approximate truth of conclusions involving generalizations. In the controlled experiments carried out using both master students and professional programmers we would like to generalize the achieved outcomes. In both the context the subjects were very representative of the population since they were either master students with J2EE knowledge or professional developers with ACUCOBOL-GT programming experience. Moreover, both the experiments were based on the same process, software tool, and the migration tasks, thus generalizing the environmental factors.

The *conclusion validity* is the most important of the four validity types because it is relevant whenever someone is trying to decide if there exists a relationship in the considered observations. A definition of conclusion validity could be: the degree to which conclusions we reach about relationships in our data are reasonable. Regarding our experiment proper tests were performed to statistically reject null hypotheses. Non-parametric tests were used in place of parametric tests where there were not the conditions necessary to use parametric tests. Also, ANOVA results were confirmed by non-parametric tests (Friedman test).

5. Conclusion and Lesson Learned

This paper presents a controlled experiment to assess the effectiveness and usefulness of a migration environment, named MELIS [16] (Migration Environment for Legacy Information Systems). MELIS is an Eclipse plug-in supporting the strategy proposed in [16] for the migration of legacy information systems implemented in

ACUCOBOL-GT towards a web based multi-tier architecture. Indeed, MELIS supports the software engineer in the migration of the graphical user interface, and in the restructuring and wrapping of the original legacy code.

Our aim consisted on one side of identifying potentially important environmental factors that could affect the results of the migration tool under investigation, thus helping to understand its external validity [7], and on the other side of investigating the effort to migrate COBOL program without having any experience on migration technologies. Regarding the migration effort, the data analysis revealed that the time to migrate a program using MELIS is on the average 17.86% of the time to migrate the same program using traditional development tools, i.e. our plug-in increases the productivity of a factor 5.6. Conversely, the controlled experiment carried out in the context of professional programmers, revealed that on average the use of MELIS increases the productivity of a factor 4 with respect to the use of traditional development tools. This difference was essentially due to the time to perform the migration tasks using the traditional software tools. In fact, to migrate the tasks using the traditional software tools the professional programmers spent less time, while the time to accomplish the tasks were comparable when MELIS was used. Thus, we can assert that MELIS partially fills the gap between programmer with COBOL competences and master students without specific experience.

The survey revealed that the master students appreciated the support provided by MELIS especially in the user interface reengineering, and integration and deployment phases.

The controlled experiment also revealed a number of directions to improve the usefulness of MELIS. A first direction would be to add some feature to further automate the restructuring of the legacy code. A second direction should aim at adding new features to better support the integration of the restructured legacy code with the reengineered graphical user interface as well as to fix some bugs identified during the deployment and use of the migrated system. Finally, some usability problems that the subjects detected and promptly annotated on the survey questionnaire will be also addressed.

Acknowledgments

The authors would like to thank Vincenzo Venezia, Giovanni Vildacci, as well as the programmers of MTSys s.r.l., our partner company, for the stimulating discussion and the precious suggestions. Special thanks are due to Marianna Borriello, Vincenzo Coppola, Rosario Di Leva, Aniello Napolitano, and Nicola Vitiello, who implemented different components of the migration environment and target architecture. Last but not least the author would like to thank Michele Mennella and Giuseppe Tagliamonte for the valuable help provided to carry out the controlled experiment.

References

- [1] ACUBENCH, Available at <http://www.acucorp.com/solutions/datasheets/acubench/>

- [2] E. Arranga and F. Coyle "Object Oriented COBOL" SIGS Books, New York, 1996.
- [3] L. Aversano, G. Canfora, A. Cimitile, A. De Lucia, "Migrating Legacy Systems to the Web: an Experience Report", *In Proceedings of the 5th European Conference on Software Maintenance and Reengineering*, Lisbon, Portugal, IEEE CS Press, 2001, pp. 148-157.
- [4] L. Aversano, G. Canfora, A. De Lucia "Migrating Legacy System to the Web: a Business Process Reengineering Oriented Approach", *Advances in Software Maintenance Management: Technologies and Solutions*, M. Polo editor, Idea Group Publishing, USA, 2003, pp. 151-181.
- [5] V.R. Basili, R.W. Selby, and D.H. Hutchens, "Experimentation in Software Engineering" *IEEE Transaction on Software Engineering*, vol. 12, no 7, 1986, pp. 733-743.
- [6] V.R. Basili, "The Role of Experimentation in Software Engineering: Past, Current, and Future" *In Proceedings of 18th International Conference on Software Engineering*, 1996, pp. 442-449.
- [7] V. Basili, F. Shull, and F. Lanubile, "Building Knowledge through Families of Experiments", *In IEEE Transactions on Software Engineering*, vol., no 4, 1999, pp. 435-437.
- [8] T. Bodhuin, E. Guardabascio, and M. Tortorella, "Migrating COBOL systems to the Web by using the MVC design pattern", *In Proceedings of the 9th Working Conference on Reverse Engineering*, Virginia, USA, IEEE CS Press, 2002, pp. 329-338.
- [9] T. Bodhuin, E. Guardabascio, and M. Tortorella. "Migration of non-decomposable software systems to the web using screen proxies", *In Proceedings of the 10th Working Conference on Reverse Engineering*, Victoria, BC, Canada, IEEE CS Press, 2003, pp.165- 174.
- [10] D. Bovenzi, G. Canfora, A. R. Fasolino. "Enabling Legacy System Accessibility by Web Heterogeneous Clients" *In Proceedings of the 7th European Conference On Software Maintenance and Reengineering*, Victoria, Canada, IEEE CS Press, pp. 73-81.
- [11] M. L. Brodie and M. Stonebraker "Migrating Legacy Systems" *Morgan Kaufmann*, San Francisco, 1995.
- [12] J. G. Butler "Mainframe to Client/Server Migration". *Computer Technology Research Corp.*, Charleston, South Caroline, 1996.
- [13] G. Canfora, A. Cimitile, A. De Lucia, and G. A. Di Lucca "Decomposing Legacy Programs: A First Step Towards Migrating to Client-Server Platforms" *Journal of Systems and Software*, vol. 54, 2000, pp. 99-110.
- [14] G. Canfora, A. Fasolino, G. Frattolillo, and P. Tramontana, "Migrating Interactive Legacy Systems to Web Services". *In Proceedings of the Conference on Software Maintenance and Reengineering*, Bari, Italy, IEEE CS Press, 2006, pp. 24-36.
- [15] W.J. Conover, *Practical Nonparametric Statistics*, 3rd Edition, Wiley, 1998.
- [16] A. De Lucia, R. Francese, G. Scanniello, G. Tortora, and N. Vitiello "A Strategy and an Eclipse Based Environment for the Migration of Legacy Systems to Multi-tier Web-based Architectures". *In Proc. of 22nd IEEE International Conference on Software Maintenance*, Philadelphia, Pennsylvania, USA, IEEE CS Press, 2006, pp. 438-447.
- [17] J. L. Devore and N. Farnum, *Applied Statistics for Engineers and Scientists*, Duxbury, 1999.
- [18] B. Endres and D. Rombach, *A Handbook of Software and Systems Engineering: Empirical Observations, Laws, and Theories*, Fraunhofer IESE series on software engineering. Pearson Education Limited, 2003.

- [19] B.A. Kitchenham, S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. Hoaglin, K. El Emam, and J. Rosenberg. "Preliminary guidelines for empirical research in software engineering", *IEEE Transaction on Software Engineering*, vol. 28, no 8, 2002, pp. 721-734.
- [20] R. Koschke. "What architects should know about reverse engineering and reengineering". *Keynote Speech in The 12th Working Conference on Reverse Engineering*, 2005 Pittsburgh, Pennsylvania, USA.
- [21] LegacyJ, http://www.legacyj.com/lgcyj_perc1.html
- [22] S. LinkMan and H. D. Rombach, "Experimentation as a Vehicle for Software Technology Transfer - A Family of Software Reading Techniques", *Information and Software Technology*, vol. 39, no. 11, 1997, pp. 777-780.
- [23] Lomboz, Available at <http://www.objectlearn.com/index.jsp>
- [24] E. Merlo, P. Y. Gagn, J. F. Gilard, K. Kontogiannis, L. Hendren, P. Panangaden and R. De Mori "Reengineering User Interfaces" *IEEE Software*, 1995, vol. 12, pp. 64-73.
- [25] M. Moore "User Interface Reengineering" *PhD Dissertation*, College of Computing, Georgia Institute of Technology, Atlanta, GA, 1998.
- [26] M. Moore, L. Moshkina. "Migrating legacy user interfaces to the internet: Shifting dialogue initiative". *In Proceedings of Working Conference on Reverse Engineering*, Brisbane, Australia, IEEE CS Press, 2000, pp. 52-58.
- [27] N. Oppenheim, "Questionnaire Design, Interviewing and Attitude Measurement", Pinter Publishers, 1992.
- [28] S. L. Pfleeger and W. Menezes, "Marketing technology to software practitioners", *IEEE Software*, vol. 17, no. 1, 2000, pp. 27-33.
- [29] M. Rahgozar and F. Oroumchian. "An effective strategy for legacy systems evolution". *Journal of Software Maintenance: Research and Practice*; vol. 15, 2003, pp. 325-344.
- [30] P. Royston, "An extension of Shapiro and Wilk's test for normality to large samples", *In Applied Statistics*, Vol. 31, no.2, 1982, pp. 115-124.
- [31] D. I.K. Sjøberg, J. E. Hannay, O. Hansen, V. B. Kampenes, A. Karahasanovic, N. Liborg, and A. C. Rekdal. "A Survey of Controlled Experiments in Software Engineering", *IEEE Transaction on Software Engineering*, vol. 31, no. 9, pp. 733-753.
- [32] H. M. Sneed "Encapsulating Legacy Software for Use in Client/Server Systems". *In Proceedings of Working Conference on Reverse Engineering*, Monterey, CA, 1996, IEEE CS Press, pp. 104-119.
- [33] H. M. Sneed. "Program interface reengineering for wrapping". *In Proc. of Working Conference on Reverse Engineering*, Amsterdam, Holland, 1997, pp. 206-214.
- [34] H. M. Sneed "Risks Involved in Reengineering Projects", *In Proceedings of the 6th IEEE Working Conference on Reverse Engineering*, Atlanta, GA, IEEE CS Press, 1999, pp. 204-211.
- [35] H. M. Sneed. "Wrapping Legacy COBOL Programs behind an XML-Interface", *In Proceedings of Working Conference on Reverse Engineering*, IEEE CS Press, 2001, pp. 189-197.
- [36] H.M. Sneed and S.H. Sneed. "Creating Web services from legacy host programs", *In Proceedings of 5th International Workshop on Web Site Evolution*, Amsterdam, The Netherlands, IEEE CS Press, 2003, pp. 59-65

- [37] H. M. Sneed. "Integrating legacy Software into a Service oriented Architecture". *In Proceedings of the Conference on Software Maintenance and Reengineering*, Bari, Italy, IEEE CS Press, 2006, pp. 3-14.
- [38] E. Stroulia, M. El-Ramly, P. Iglinski, and Paul Sorenson. "User Interface Reverse Engineering in Support of Interface Migration to the Web" *Automated Software Engineering*, vol. 3, no. 10, Kluwer Academic Publishers, 2003, pp. 271-301.
- [39] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell and A. Wesslen, *Experimentation in Software Engineering – An Introduction*, Kluwer, 2000.
- [40] M. V. Zelkowitz, and D. R. Wallace, "Experimental Models for Validating Technology", *IEEE Computer*, May 1998, pp. 23-31.