

Introducing Legacy System Migration Technologies in a Software Company: a Controlled Experiment

Massimo Colosimo[◦], Andrea De Lucia[◦], Rita Francese[◦], Giuseppe Scanniello^{*}, and Genoveffa Tortora[◦]

[◦]*Dipartimento di Matematica e Informatica, University of Salerno, Fisciano (SA), ITALY*

^{*}*Dipartimento di Matematica e Informatica, University of Basilicata, Potenza, ITALY*

cillo_81@yahoo.it, {adelucia, francese}@unisa.it, giuseppe.scanniello@unibas.it, tortora@unisa.it

Abstract

Transferring reverse engineering and migration technologies to industry requires to evaluate whether they potentially fulfill industry needs. We present a controlled experiment aimed at assessing the usefulness of a migration tool, named MELIS (Migration Environment for Legacy Information Systems) in an industrial setting. MELIS is an Eclipse plug-in conceived to migrate legacy information systems to a web-enabled multi-tier target architecture. This plug-in supports the software engineer in the migration of the graphical user interface and in the restructuring and wrapping of the original legacy code. The context of the experiment was constituted of professional programmers, with COBOL programming experience, and academic subjects, with J2EE programming experience. The results revealed that on average the use of MELIS increases the productivity of a factor 4 with respect to the use of traditional development tools.

1. Introduction

Empirical software engineering helps determine the effectiveness of proposed theories and methods. A huge number of researches in have been presented to determine whether a particular technique is effective [5][6][17][20][21][22][33][43]. Research in the empirical software engineering field should aim at acquiring general knowledge about *which* process, method, technique, language, or tool is useful for *whom* to conduct *which* tasks in *which* environments [33]. Guidelines to define, plan and conduct empirical studies have also been defined [43][20][22].

Empirical studies are particularly important to transfer software engineering technologies produced in research laboratories to practitioners [25][31], to evaluate whether they potentially fulfill the industry needs. Their effective adoption cannot be made without a systematic and quantitative evaluation. If a tool is difficult to use, it will hardly be adopted, no matter how useful it may be. Different types of empirical studies can be conducted, including surveys, controlled experiments, and case studies [43]. An important category is that of the controlled experiments, which represents the classical scientific method for identifying cause-effect relationships [33].

In this report we present a controlled experiment aimed at evaluating the usefulness of introducing a legacy system migration tool within a software company. Legacy Information Systems (LISs) are written in some legacy language like COBOL and do not directly interoperate with other applications. Generally LISs are business-critical and operate 24 hours a day [7][37]. Their technical infrastructure is obsolete and does not interact with the Web. To remain competitive in the global market, organizations are forced to migrate their LISs to modern web architectures [3][4][14][29][39][40].

The context of the experiment is a research project funded by MTSys s.r.l., an Italian software company. This company developed, evolved, and marketed over the last 30 years COBOL software systems that underwent many migration waves, including mainframe to PC downsizing, COBOL dialect porting, character-based to graphical user interface migration. Currently, these systems are written in ACUCOBOL-GT and run on MS-Windows machines. The goal of the research project is to develop methods and tools to migrate these legacy systems to web technologies.

Based on the technical assessment of the legacy systems, we defined a migration strategy and developed a tool, named MELIS (Migration Environment for Legacy Information Systems), to support the phases of the migration process, and evaluated it in a case study conducted on one of the legacy systems of the partner company [15]. In particular, MELIS supports the software engineer in the migration of the graphical user interface, and in the restructuring and wrapping of the original legacy code.

In this report we evaluate the effectiveness of using MELIS for legacy system migration with respect to traditional development environments used within the software company. The controlled experiment was conducted within the laboratories of the partner company and carried out by four employees (two seniors and two young) of the company with COBOL programming experiences and four academics (graduate students and young researchers), with J2EE programming experiences. The results revealed that on average the use of MELIS increases the productivity of a factor 4 with respect to the use of traditional development tools.

2. The Migration Environment

The assessment of the legacy systems of the subject company revealed that these systems have a very low degree of decomposability [11]. For this reason we adopted a migration strategy aiming at reengineering the user interface using web technology, turning interactive legacy programs into batch programs and encapsulating them using a

communication wrapping layer, enabling the new user interface to interoperate with the restructured and wrapped legacy program. This 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][37].

Figure 1 shows the target architecture of the migration strategy. The *Wrapper* component acts as communication middleware and enables 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 Screen Sections of the original 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 through the wrapper. 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). The GUI deliverer is also responsible

The Wrapper is a generic component that is developed once for all and can be used in the migration of any LISs. 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 starts 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 can be found in [15].

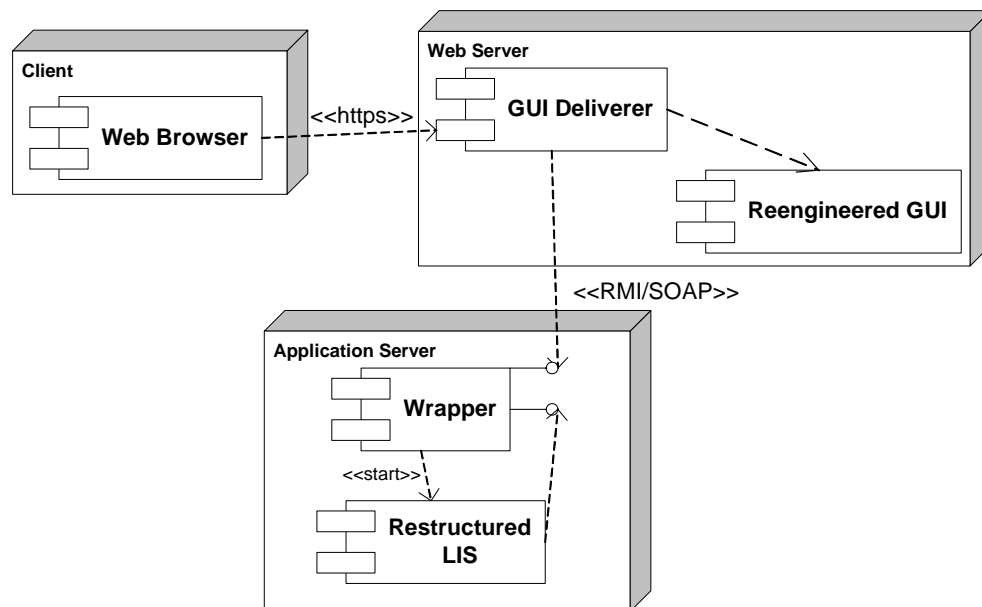


Figure 1. The target software architecture

Figure 2 shows the legacy system migration process supported by MELIS: rounded rectangles represent the phases of the process and rectangles the intermediate produced artifacts. The *Pre-processing* phase 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 SECTION is encoded in a XML file. In this phase the BEFORE and AFTER clauses are classified as: format check operations, which are migrated to the client side, and field database access or application logic operations, which are left on the restructured COBOL program and invoked using the wrapper.

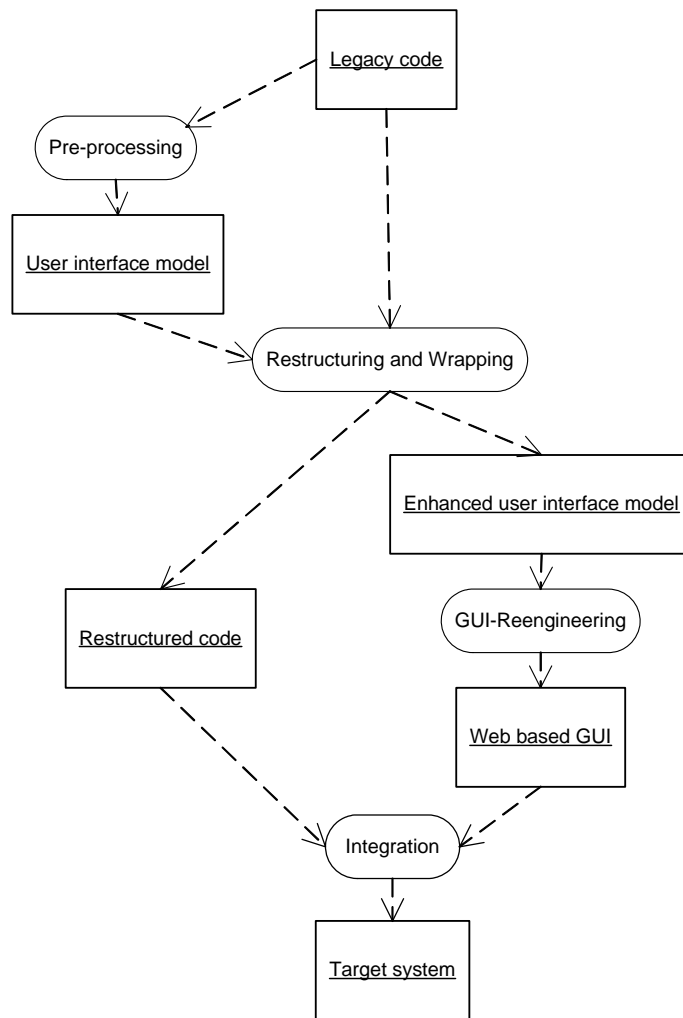


Figure 2. The migration process

MELIS automatically generates the XML file, which includes the actions associated with a SCREEN SECTION performed when the entry fields are filled in and the confirm button is pushed. Moreover, for each identified

SCREEN SECTION the XML file contains details of the corresponding graphical objects and the associated controls. The XML file is used to hierarchically organize the code associated to the SCREEN SECTIONs. 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. Javascript language is used to migrate on the client side checks that do not need any interaction with the LIS, as for example the validation of the date format. MELIS also proposes a library containing a set of javascript functions that can be reused during the client-side control migration. The source code of checks involving database accesses cannot be migrated on the client. To support the developers, MELIS also analyzes the original legacy code and highlights it in yellow when the checks can be migrated on the client side, in red otherwise.

During the Restructuring and Wrapping phase the LIS code is restructured to become a batch program. To this aim MELIS automatically comments (through a “*”) the statements DISPLAY and ACCEPT 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 dynamic web pages for each subsystem to migrate. The web based interfaces of the migrated application is built from the intermediate SCREEN SECTION representation. The software engineer uses MELIS to generate a JSP page (see Figure 3) for each SCREEN SECTION of a given ACUCOBOL-GT subsystem. To enable the communication between each JSP page and the restructured legacy code both a servlet and a Java bean are also generated. In particular, the Java bean contains the fields of the SCREEN SECTIONs and is built from the XML description contained in the enhanced user interface model. The generated pages are very similar to the original one, but they can be eventually modified or enhanced by using the MELIS authoring feature. For example, Figure 3 shows a JSP page generated by MELIS and then enhanced to exactly reproduce the layout of the original graphical user interface. Moreover, MELIS also adopt Cascading Style Sheets (CSS) 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.

Finally, the *Integration* phase has to be carried out in order to integrate 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 integrated in MELIS and then deploys the reengineered web based graphical user interfaces (i.e., the generated beans, servlets, and dynamic pages) on the *Web Server* node and the compiled code of restructured legacy program on the *Application Server* node.

It is worth pointing out that the effort required to migrate a LIS is mainly concentrated on the *GUI Reengineering* and *Restructuring and Wrapping* phases due to the embedded control flow in the original graphical user interface.

3. The Controlled Experiment

In this section we describe the proposed controlled experiment, which is described adopting the template proposed by Wohlin *et al.* [43].

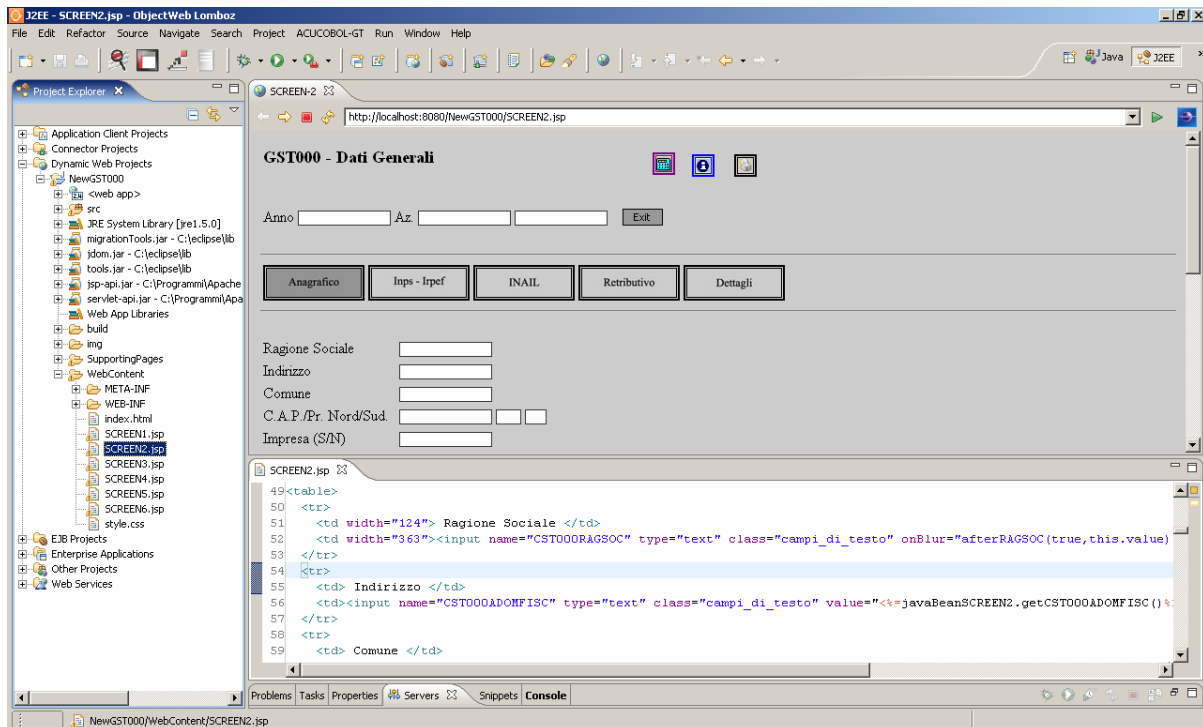


Figure 3. MELIS snapshot

3.1 Experiment definition and context

The controlled experiment aimed at analyzing and evaluating the usefulness of MELIS in the migration of LIS implemented in ACUCOBOL-GT towards a web based multi-tier architecture and the possibility of adopting MELIS within the partner software company.

The subjects of the experiment were eight volunteers, and in particular four employees of the partner company, with COBOL programming experiences, and 4 academic subjects recruited at University of Salerno, with J2EE programming experiences. The subjects' background and the corresponding role within the organizations are summarized in Table 1. It is worth pointing out that neither the industrial subjects, nor the academic subjects were familiar with MELIS and the supported migration strategy.

The subjects were randomly grouped in 4 development teams each composed of a software company employee and an academic subject, as shown in Table 2. This table also reports the hardware configurations of the PCs that the recruited subjects used during the controlled experiment.

The experiment has been performed online, i.e., in a laboratory of the industrial partner. The development teams had to perform the following two tasks:

- **T₁**: migrating an ACUCOBOL-GT program computing the basic arithmetic operations, i.e. addition, subtraction, multiplication and division;
- **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.

The two tasks propose the majority of the problems that a developer could encounter in the migration of an ACUCOBOL-GT program. The tasks have been chosen to accomplish each laboratory session within 3 hours and a half. 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.

Table 1. Recruited Subjects

ID	Role within the organization	COBOL Experience	J2EE Experience
M1	Manager	more than 10 years	none
M2	Employee	1 year	none
M3	Employee	1 year	none
M4	Employee	more than 10 years	none
U1	Ph D Student	None	between 1 and 5 years
U2	Ph D Student	none	between 1 and 5 years
U3	Research Fellow	none	less than 1 year
U4	Master student	none	between 1 and 5 years

Two different migration methods 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 the first method, namely *PL*, is performed using the MELIS Eclipse plug-in, while the second method, namely *NOPL*, is performed using the software development environments known to the subjects involved in the experiment. These environments are ACUBENCH [1], an

ACUCOBOL-GT environment (used for restructuring and wrapping the COBOL programs), and Lomboz [26], an Eclipse plug-in for developing J2EE applications (used for reengineering the user interface).

Table 2. Composed groups

Group	Industrial Employees	Academic Subjects	PC
A	M2	U3	Intel Centrino 1,6 GHz RAM 512
B	M1	U1	Intel Pentium III 1 GHz RAM 512
C	M3	U4	Intel Centrino 1,5 GHz RAM 512
D	M4	U2	Intel Pentium IV 1,4 GHz RAM 512

3.2 Hypothesis

As mentioned before the proposed controlled experiment aim at assessing whether the use of MELIS during the migration tasks helps to improve the developers' productivity. Consequently, the following null hypothesis has been formulated:

- **H₀₁**: 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;

It worth pointing out that the effort to migrate an ACUCOBOL-GT program to the web includes both its comprehension and the times to accomplish all the phases of the migration process.

3.3 Selected variables and experiment design

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

- **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.

Differently, the dependent variable is the time required to migrate the assigned tasks.

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 plug-in on both tasks. The design of the experiment is summarized in Table 3.

Table 3. 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

3.4 Preparation

We installed MELIS, ACUBENCH [1], and Lomboz [26] on each PC involved in the controlled experiment. A training session has been carried out to give subjects an equal prior knowledge of the software applications, target architecture, and migration process. We also introduced MELIS to let subjects get confidence with it through some examples (not related to the tasks to avoid biasing the experiment). The training session was concluded presenting detailed instructions on the tasks to be performed. 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 4 has been presented. The questionnaire is composed of questions expecting closed answers according to the Likert scale [30]: from 1 (strongly agree) to 5 (strongly disagree).

3.5 Material and execution

For each laboratory session, the development teams carried out their tasks without time limit. Once the each task was accomplished the teams filled in the survey questionnaire, while the supervisors collected the 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.

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 treatment, 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 questionnaire.

3.6 Experiment results

After the execution of the experiment, we collected the times to accomplish the laboratory sessions of each development team. In particular, Table 5 details the times expressed in minutes, grouped by *laboratory sessions*, *Treatment*, and *Team*. On the other hand, the box plot in Figure 4 summarizes the achieved times according to the considered *Treatments*, while Table 6 presents the descriptive statistics of the experiment. The development teams spent on the average 48.75 minutes to migrate the assigned ACUCOBOL-GT programs using MELIS, while they spent on average 191.5 minutes without using it. Therefore, the time to migrate a program using MELIS is on the average 25% of the time to migrate the same program without the MELIS usage.

Table 4. 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%)?

Table 5. Raw data expressed in minutes

	Treatment	Team A	Treatment	Team B	Treatment	Team C	Treatment	Team D
Lab 1	T1_PL	56	T1_NOPL	184	T2_PL	69	T2_NOPL	180
Lab 2	T2_NOPL	197	T2_PL	40	T1_NOPL	205	T1_PL	30

To assess that the accomplishment of the migration tasks is not influenced by the technical competences of the recruited subjects we used both Spearman's rho [41] and Pearson [18] correlations. These tests revealed that the subjects' technical competences are not statistically correlated with the effort to perform the migration tasks.

A two-way Analysis of Variance (ANOVA) [16] was used to investigate the effect of the selected dependent variable by the investigated factors (i.e. the performed task and the use of MELIS). 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.

Table 6. Descriptive statistics

Task	Method	Mean	Median	Std. Dev
All	PL	48.75	48	17.23127
	NOPL	191.5	190.5	11.56143
T1	PL	43	43	18.38478
	NOPL	194.5	194.5	14.84924
T2	PL	54.5	54.5	20.5061
	NOPL	188.5	188.5	12.02082

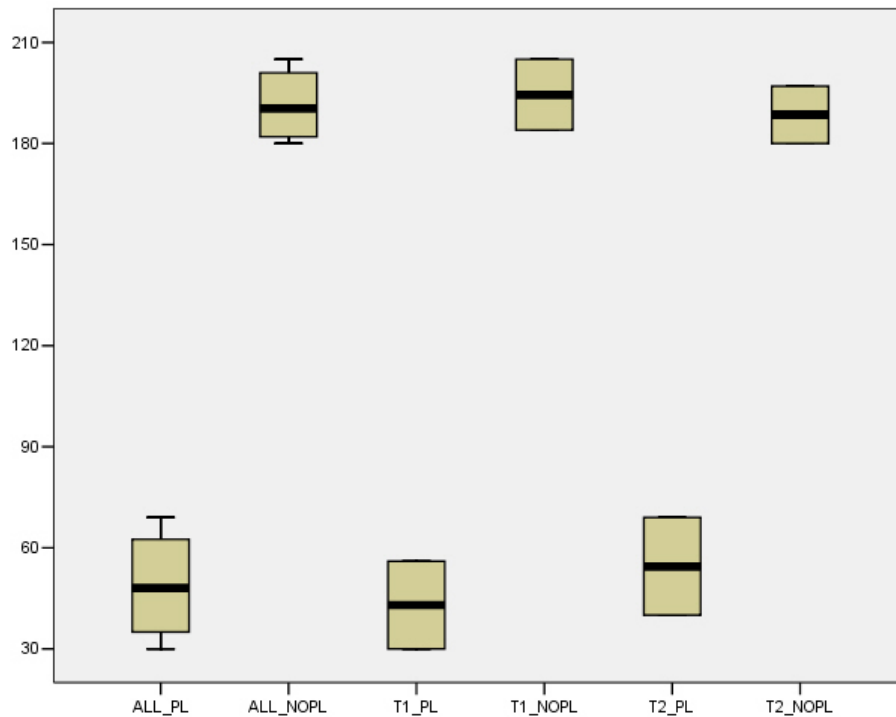


Figure 4. Box plots of the tasks' times

Table 7 summarizes the results of the two-way ANOVA test. We first observe that the interaction between *Method* and *Task* is not significant. There is very highly significant (Sig. = 0.000) effect of the *Method* overall, while the *Task* factor is not significant (Sig. = 0.828). 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.

The data collected from the survey questionnaire are visually summarized by the boxplots in Figure 5. The answers of the survey questionnaire corresponding to the tasks T1 and T2 are summarized by the boxplots on the left and on the right hand sides, respectively. The analyses of the collected data showed that the time to perform both the tasks was considered appropriate. Although the task objectives were generally considered clear, the objectives of the task T2 (i.e. the migration of the ACUCOBOL-GT program with embedded controls) were less clear. Moreover, the task T2 was also considered less clear than the task T1 as the boxes corresponding to the question Q3 show. This difference was probably due to the different complexity degree of the assigned tasks.

The information provided by hard copy material was generally considered adequate to perform the assigned tasks. The agreement level concerning the question Q5 is appropriate as the medians of the corresponding boxplots reveal. On the other hand, the reengineering of the original graphical user interface of the task T1 was considered simpler than T2. This difference is due to the embedded controls within the SCREENSECTION of the task T2. Finally, the agreement level regarding the integration and the deployment of the migrated programs of the tasks T1 and T2 was concordant and adequate.

Table 7. ANOVA table of similarity by Method and Task ($R^2 = 97,3\%$, $R^2(\text{adj}) = 95,3\%$)

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	40,923.375	3	13,641.125	48.567	0.001
Intercept	115,440.125	1	115,440.125	411.002	0.000
Method (row)	40,755.125	1	40,755.125	145.101	0.000
Task (Column)	15.125	1	15.125	0.054	0.828
Interaction	153.125	1	153.125	0.545	0.501
Residual	1,123.500	4	280.875		
Total	157,487.000	8			
Corrected Total	42,046.875	7			

3.7 Threats to validity

The threats to validity that could affect the experiment (i.e. internal, construct, external, and conclusions validity threats) are described in this section.

Generally, the *internal validity* is only relevant in studies that try to establish a causal relationship. Thus, 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 and the target architecture. The key question in internal validity was whether observed changes can be attributed to the learning effect and not to other possible causes. Concerning our experiment the *internal validity* threats are mitigated by the experiment design, since each group worked, over the two *Labs*, on different *Tasks* and with two different *Methods* (*PL* and *NOPL*). This result was also confirmed by the two-way ANOVA test, which did not reveal a significant learning effect of the subjects across the laboratory sessions. The survey also confirmed that the subjects found clear everything regarding the tasks of the experiment. Moreover, the subjects did not know exactly the hypothesis of the experiment although they applied the migration strategy and adopted the technological infrastructure of the target architecture. It is also worth noting that subjects were not evaluated on their performance within the laboratory sessions.

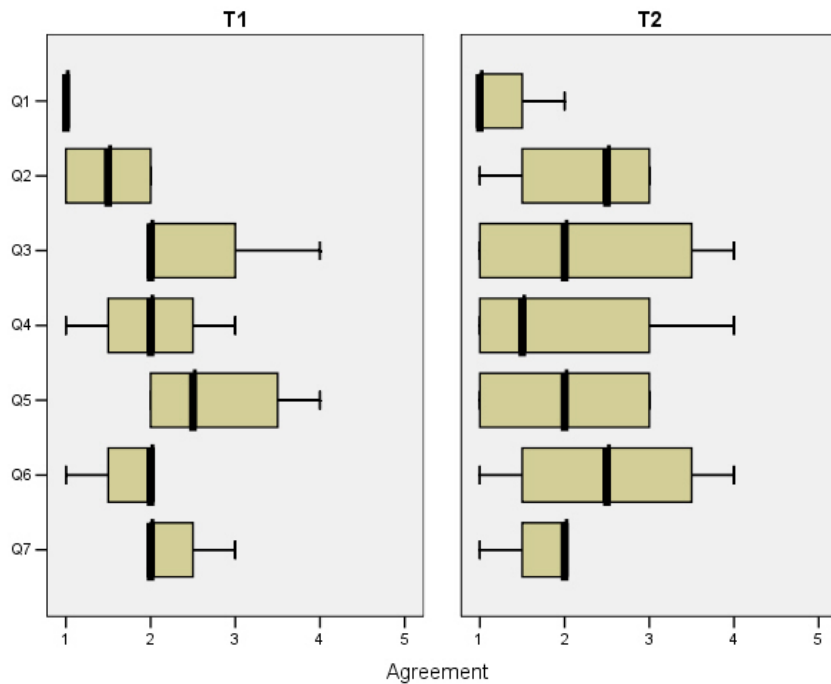


Figure 5. Boxplots of the survey questionnaire

The *construct validity* threats that could be present in this experiment, i.e. the interactions between different treatments, were mitigated by a proper design that allowed separating the analysis of the different factors and of their interactions. Moreover, 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 experiment supervisors. Finally, the survey questionnaire was designed using standard ways and scales [30].

External validity refers to the approximate truth of conclusions involving generalizations. To select the subjects of our experiment we first identified the population we would like to generalize the MELIS usage. After that we drew a fair sample from that population and conducted our experiment with subjects belonging to this sample. We can assert that the selected sample was very representative of the population. In fact, subjects were developers of our industrial partner, while the academic subjects had a very good analysis, development and programming experience. Due to their experience academic subjects are not far from industry programmers. Nevertheless, it is worth replicating the experiment with J2EE industry developers and other migration environments, to confirm or contradict the obtained results.

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).

4. Related Work

Several methods and tool have been presented in the literature for the migration of legacy systems [10][11][12][13][32][36][38], and recently for the migration to web technologies [3][4] [9][14][29][39].

Rahgozar and Oroumchian [32] first proposed a classification of the current approaches to evolve LISs and then suggested guidelines and a methodology for the migration of transaction oriented systems. The authors also proposed 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.

Several authors have addressed the reengineering of user interfaces. In particular, approaches and methodologies based on data-flow analysis [27] and State Transition Diagrams [10][42] as well as on techniques derived from artificial intelligence [28] have been proposed. Moore and Moshkina [29] 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 [36]. 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. The wrapper is designed to satisfy service stability, data integration, and application integration requirements. The authors proposed 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].

Technical aspects to decompose LISs into client and server components have been investigated [12][13][34]. Generally, wrapping and reverse engineering of software architectures [23] have a relevant role in the migration of LISs [2][3][4][8][9][35]. In particular, wrapper interfaces can be used by legacy or ex-novo software components to use the encapsulated components in a transparent way. In addition, there are some commercial solutions [19][24] permitting LISs to be integrated with the modern distributed technology.

Sneed [38] proposed tools and wrapping techniques. Online programs were transformed into data driven subprograms which process an XML-document. Batch programs were adapted to read and write XML-documents. Finally, in subprograms parameters were set from an XML-document. Recently, Sneed in [40] has been extended this approach to enable the integration of a legacy system into a Service Oriented Architecture.

Several approaches to migrate LISs towards the web have also been proposed [3][4][8][9][39]. 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 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 [24]. 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.

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

5. Conclusion and lesson learned

In this report we have presented a controlled experiment to assess the effectiveness and usefulness of a migration environment, named MELIS [15] (Migration Environment for Legacy Information Systems). MELIS is an Eclipse plug-in supporting the strategy proposed in [15] 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.

The data analysis collected from the controlled experiment allowed us to establish that MELIS increases the productivity. In particular, we observed that using our plug-in the time to migrate an ACUCOBOL-GT program decreases of 75%. We expect that the productivity will be further improved on more complex ACUCOBOL-GT programs, as the case study presented in [15] suggests. In this case study we migrated a meaningful subsystem of 6691 LOCs, which was restructured in a program of 8268 LOCs. In particular, MELIS added 1447 LOCs, while 130 LOCs were manually coded by the developers, who employed 39 person/hours to migrate the whole subsystem. In this case the developers were familiar with MELIS and the adopted technologies.

We also observed that the competences of the recruited subjects marginally influenced the effort to reengineer the graphical user interfaces. It is mainly due to the fact that MELIS generates a web based graphical user interface similar to the original one. Hence, the development team can avoid to modify the graphical user interface, when not explicitly required. The restructuring of the legacy code requires more effort since this phase is not fully automated. As a result, developers with COBOL experience spent less time to restructure the original source code.

The controlled experiment also revealed a number of directions to improve the usefulness of MELIS. A first direction would be to add new features to better support the integration of the restructured legacy code with the reengineered graphical user interface as well as fixing some bugs identified during the deployment and use of the migrated system. A second direction concerns fixing some usability problems that the subjects detected and promptly annotated on the survey questionnaire. Finally, we are going to replicate the controlled experiment with a larger dataset.

Acknowledgements

The authors would like to thank Vincenzo Venezia, Giovanni Vildacci, Pasquale Crescitelli and Giulio Iannella of MTSYS srl, for the participation to the controlled experiment. Special thanks are also due to Fausto Fasano, Rocco Oliveto, Michele Risi, and Alessandro Arcella, who were involved in the experiment as academic subjects. The authors are also grateful to Marianna Borriello, Vincenzo Coppola, Rosario Di Leva, Aniello Napolitano, and Nicola Vitiello who implemented some MELIS components.

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] K. Bennett, M. Ramage, and M. Munro, “Decision model for legacy systems” *In IEE Proceedings Software*, 146(3), 1999, pp. 153-159.
- [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] 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, 2006, (to appear).
- [16] J. L. Devore and N. Farnum, *Applied Statistics for Engineers and Scientists*, Duxbury, 1999.

- [17] A. 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.
- [18] J.E. Freund. "Mathematical statistics". In *5th ed. Upper Saddle River, NJ: Prentice Hall, 1992*; pp. 494-546.
- [19] IBM WebSphere software: Legacy modernization with WebSphere Studio Enterprise Developer, 2002. At <http://www.redbooks.ibm.com/redbooks/pdfs/sg246806.pdf>.
- [20] N. Juristo, A.M. Moreno, and S. Vegas, "Reviewing 25 Years of Testing Technique Experiments", *Empirical Software Engineering*, vol. 9, 2004, pp. 7-44.
- [21] C.F. Kemerer and S.Slaughter. "An Empirical Approach to Studying Software Evolution", *IEEE Transaction on Software Engineering*, vol. 25, no 4, 1999, pp. 493-509.
- [22] 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.
- [23] 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.
- [24] LegacyJ, http://www.legacyj.com/lgcyj_perc1.html
- [25] 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.
- [26] Lomboz, Available at <http://www.objectlearn.com/index.jsp>
- [27] 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.
- [28] M. Moore "User Interface Reengineering" *PhD Dissertation*, College of Computing, Georgia Institute of Technology, Atlanta, GA, 1998.
- [29] 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.
- [30] A. N. Oppenheim, *Questionnaire Design, Interviewing and Attitude Measurement*, Pinter Publishers, 1992.
- [31] S. L. Pfleeger and W. Menezes, "Marketing technology to software practitioners", *IEEE Software*, vol. 17, no. 1, 2000, pp. 27-33.
- [32] 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.
- [33] 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.
- [34] H. M. Sneed and E. Nyary. "Downsizing Large Application Programs", *Journal of Software Maintenance: Research and Practice*, vol. 6, no. 5, 1994, pp. 105-116.
- [35] 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.
- [36] H. M. Sneed. "Program interface reengineering for wrapping". In *Proc. of Working Conference on Reverse Engineering*, Amsterdam, Holland, 1997, pp. 206-214.
- [37] 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.

- [38] 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.
- [39] 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
- [40] 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.
- [41] C. Spearman "The proof and measurement of association between two things". *American Journal of Psychology*, no.15, 1904, pp. 72-101.
- [42] 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.
- [43] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell and A. Wesslen, *Experimentation in Software Engineering – An Introduction*, Kluwer, 2000.
- [44] M. V. Zelkowitz, and D. R. Wallace, "Experimental Models for Validating Technology", *IEEE Computer*, May 1998, pp. 23-31.