Noname manuscript No.
(will be inserted by the editor)

Empirical study on the maintainability of Web applications: Model-driven Engineering vs Code-centric

Y. Martínez · C. Cachero · S. Meliá

Received: date / Accepted: date

Abstract BACKGROUND: Model-driven Engineering (MDE) approaches are often acknowledged to improve the maintainability of the resulting applications. However, there is a scarcity of empirical evidence that backs their claimed benefits and limitations with respect to code-centric approaches.
OBJECTIVE: To compare the performance and satisfaction of junior software maintainers while executing maintainability tasks on Web applications with two different development approaches, one being OOH4RIA, a model-driven approach, and the other being a code-centric approach based on Visual Studio .NET and the Agile Unified Process.
METHOD: We have conducted a quasi-experiment with 27 graduated students from the University of Alicante. They were aleatory divided into two groups, and each group was assigned to a different Web application on which they performed a set of maintainability tasks.
RESULTS: Maintaining Web applications with OOH4RIA clearly improves the performance of subjects. It also tips the satisfaction balance in favor of OOH4RIA, although not significantly.
CONCLUSIONS: Model-driven development methods seem to improve both the developers' objective performance and subjective opinions on ease of use and utility of the method. Further experimentation is needed to be able to generalize the results to different populations, methods, languages and tools, different domains and different application sizes.

Y. Martínez
Universidad Máximo Gómez Báez de Ciego de Ávila, Cuba
E-mail: yulkeidi@gmail.com

C. Cachero
DLSI. Universidad de Alicante, Spain
E-mail: ccachero@dlsi.ua.es

S. Meliá
DLSI. Universidad de Alicante, Spain
E-mail: santi@dlsi.ua.es

Keywords Maintainability, Satisfaction, Quasi-experiment, MDE

1 Introduction

Model-driven Engineering (MDE) is a software development paradigm that heavily relies on models and model transformations for the development, maintenance and evolution of software [42] [50] [44]. The MDE community claims several advantages over traditional development (code-centric) approaches, among which (a) short and long term productivity gains, (b) improved project communication and (c) improved quality of the resulting application are specially relevant [34] [22] [24]. The reason for such claims is that current MDE approaches offer high-level abstractions that capture some of the most salient characteristics of modern applications. Such abstractions speed up the definition of models that can be used to generate applications for different implementation environments.

In particular, software maintenance is one of the areas in which MDE claims to be able to make a greater impact in terms of both reliability and efficiency [6]. Software maintenance is defined in the IEEE 1219 Standard [28] as “the modification of a software product after being delivered to correct faults, improve performance or other attributes, or to adapt the product to a modified environment”. More concisely, the ISO/IEC 9126 standard [1] states that maintainability is “the capability of the software product to be modified”[1]. Software maintenance is important because: (a) it consumes between 45% and 60% of the overall life cycle costs and, (b) the inability to change software quickly and reliably means that business opportunities may be lost [43].

However, practitioners still lack a body of practical evidence that soundly backs the maintainability advantages claimed by the MDE community [24,45]. This contrasts with other disciplines and even other areas of Software Engineering (SE) in which empirical evidence is common [62]. Without empirical evidence, there is a danger for resources to be wasted and for software tools to fail to develop appropriately [25]. Such evidence can be provided in the shape of empirical studies, which are crucial to the evaluation of processes and human-based activities. The empirical studies can be classified into surveys, experiments (be them true experiments or quasi-experiments), case studies and postmortem analyses [61]. Experimentation, specifically, provides a systematic, disciplined, quantifiable and controlled way of evaluating human-based activities.

In an effort to increase the empirical evidence regarding the impact of MDE approaches on maintainability, this paper presents a quasi-experiment in which two groups of junior developers have been asked to perform a set of maintainability tasks on two Web applications with two different approaches: one code-centric and one model-driven.

The paper is structured as follows: Section 2 characterizes the maintainability concept. Both the maintainability characterization and the empirical evidence found

[1] Although this standard is now superseded by ISO/IEC 25010 [29], the publication of the latter took place one month after the experiment went through its data gathering phase. For this reason, this paper sticks to the ISO 9126 definitions and characterizations.

in literature set the context for the definition of the experimental design (context, planning, operation and data collection mechanisms) in Section 3. Section 4 presents the data analysis and an interpretation of results that takes into account the identified threats to validity. Section 5 presents the existing empirical evidence regarding how maintainability is impacted by the adoption of MDE methods, and discusses the relationship and results of this experiment with respect to a previous one by the same authors. Last, Section 6 concludes the paper and presents some further lines of research.

2 Maintainability concepts

Maintainability tasks can be subclassified into four different types [11]:

- *Corrections*: Corrective maintainability refers to the capability to detect errors, diagnose the problems and fix them.
- *Improvements*: Perfective maintainability refers to the capability to extend the software according to new requirements or enhancements.
- *Adaptations*: Adaptive maintainability refers to the capability to modify the software in order to cope with the effects of environmental changes.
- *Preventions*: Preventive maintainability refers to the capability to support internal reengineering processes without adverse impact.

Out of them, corrections and improvements are the two most common types of maintainability tasks in software development [45], and therefore the two types of tasks that most heavily influence the global maintainability of software applications.

Furthermore, regardless of the type of maintainability task, the ISO/IEC 9126 standard [1] establishes that five different sub-characteristics must be taken into account in order to assess the maintainability of software applications:

- *Analysability*: Capability of the software product to be diagnosed for deficiencies or causes of failure in the software, or for the parts to be modified to be identified.
- *Changeability*: Capability of the software product to enable the application of a specified modification.
- *Stability*: Capability of the software product to avoid unexpected effects from modifications of the software.
- *Testability*: Capability of the software product to enable modified software to be validated.
- *Compliance*: Capability of the software product to meet the standards or conventions relating to maintainability.

3 Description of the Experiment

In February 2011, a quasi-experiment was conducted at the University of Alicante. A quasi-experiment is a type of controlled experiment in which individuals or teams (the study units) engage in one or more tasks for the sake of comparing different processes, methods, techniques, languages or tools (the treatments) [30]. In quasi-experiments subjects are not randomly chosen but rather selected on the basis of certain criteria. In our study, we have selected the subjects that decided to enroll for a Master's degree. Many authors have written about the importance of providing and correctly reporting empirical evidence in SE [17,32,31]. Quasi-experiments, although suffering from a lower internal validity than true experiments, are widely used and deemed useful in the Empirical SE field, since they allow investigations of cause-effect relations in settings such as ours, in which randomization is too costly [30].

3.1 Goals and Context Definition

Following the GQM template [53], this empirical study is aimed *at analyzing* OOH4RIA and .NET (which, in this paper, stands for the combination of the *C#* language, the .NET framework and the Visual Studio IDE) *for the purpose of* evaluating model-driven against code-centric changeability practices *with respect to* their performance and satisfaction *from the point of view of* junior developers. *The context of the study* is graduate students enrolled for the "Master in Web Applications Development" at the University of Alicante, where both a code-centric Web development process (based on *C#*, the .NET framework and the Visual Studio IDE) and a model-driven Web development process (based on the OOH4RIA method and the OOH4RIA IDE) are taught.

The description and rationale behind this selection of tools and languages will be explained in section 3.1.4. We are conscious that the selection of specific tools and languages diminishes the external validity of the study, that is, the generalizability of the results. Such a selection was necessary, nevertheless, in order to provide a real working environment, and thereby increase the construct validity of the experiment. This issue will be further discussed in Section 4.6.

The focus on changeability is due to the fact that, specially in MDE environments, not all the maintainability sub-characteristics are equally critical. Provided that the chosen MDE approach is mature enough (such as is the case of OOH4RIA), we can safely assume testability and compliance: in an automated code generation environment these issues are borne in mind during the code generation process. Also, OOH4RIA provides many clues about possible errors and side effects -a facility that is not available for .NET-, which may the comparability of any kind of analyzability or stability measures.

The design of the experiment is based on a well-known framework for experimentation in SE research [61]. Also, a laboratory package has been compiled for the sake of replicability [37].

The research questions addressed in this study have been formulated as follows:

– RQ1: How does the changeability performance (objective efficiency and objective effectiveness) of OOH4RIA compare with respect to the changeability performance of .NET?
– RQ2: How does the changeability satisfaction (perceived usefulness and perceived ease of use) of OOH4RIA compare with respect to the changeability satisfaction of .NET?

All the questions have been devised to be answered by quantitative means.

3.1.1 Subjects

The experimental subjects of our study were 30 students enrolled for the "Master in Web Applications Development" at the University of Alicante during the year 2010-2011. Two students out of those 30 abandoned the master before the experiment took place, and another one did not attend the course the day in which the experiment was scheduled for justified reasons. Since the abandonment of the experiment had nothing to do with the experiment, we can assume that the results of the experiments have not been compromised. The final sample comprised 26 men and 1 woman, of whom 20 had more than 2 years of experience developing Web applications. The mean age of the participants was 25.6 years old and all of them were Computer Engineering graduates of the University of Alicante.

Regarding the subjects' level of knowledge with respect to the different technologies and methods used during the experiment, a pretest questionnaire showed that the subjects had no previous practical knowledge of MDE, although 15 of them were aware of the existence of the paradigm. This notwithstanding, 22 knew UML (the standard on which the OOH4RIA method is based), and, from them, 3 considered that they had a high-level of knowledge of UML. It should also be noted that 20 of the subjects had previously programmed with .NET during their degree course, although only 3 had applied it in industry. By the time the experiment took place, the subjects had received additional training both in .NET and OOH4RIA. It consisted in 30 hours of programming in *C#* using Visual Studio 2010 and 30 hours of modeling with UML and the OOH4RIA tool.

3.1.2 Sensitivity analysis of the design

Since our sample size was fixed, and came determined by the number of students enrolled fir the master degree, we decided to conduct a sensitivity analysis before going on with the experiment. The objective of this analysis was to make sure that the analyses had enough power as to limit the risk of a Type II error (accepting the null hypothesis when it is in fact false). Cohen [13] suggests that the power of an analysis should be greater or equal to 0.7 to be of use. This means a maximum of a 30% chance of failing to detect an effect that is actually there.

A sensitivity analysis of all these parameters (alpha=0.05, desired power=0.70, total sample size=27) yielded a detectable effect size for the method variable of 0.25. This means that this experiment is able to detect significant differences between the two methods 70% of the times as long as the method used accounts for at least 25% of the overall (effect+error) variability. This value is consistent with the work of Cohen [13] and the results reported in literature so far (see Section 5). Detecting significant differences with lower effect sizes are of little interest in our context, given the great investment needed to change a development method inside an organization, while much larger effects are, according to our related work research, not likely to be found. This calculation was done with the statistical tool G*power 3.1.5 [18].

3.1.3 Application

Subjects were randomly assigned to either one of the following two applications:

- A Petstore application (an adaptation of an example used in [55]). This application is a virtual store of pets (Petstore) in which a client (Client) can carry out many purchase orders (Orders). The client can add to her purchase as many order lines (OrderLines) as she needs. A pet (Article) can be associated to many order lines. Pets can be classified into categories (Category), *e.g.* Bird, and subcategories (Subcategory), *e.g.* Predator.
- A Mediaplayer application (used in [19,20]). This application is a music reproducer that allows a User (User) to manage a group of songs (Song), and to define lists of reproduction (Playlist). The application allows to organize each song according to different criteria (gender, year, etc.). Also, each song, besides the URL location, may contain information on the record company, cover, letter, duration, etc. The application also stores the artists (Artist) and the albums (Album) to which such song belongs.

For both applications, subjects performed a set of maintainability tasks [2] on the server side. Both applications share the application type (data-intensive) and the complexity, both in terms of code (for the .NET treatment) and in terms of conceptual constructs (for the MDE treatment). Regarding coding, lines of code are generally preferred over functional points to measure complexity. This is due to the higher reliability of the measure, since it does not involve human judgement [61]. Regarding modeling, classes, attributes, operations and relationships are the most common set of measures used to evaluate complexity.

The characterization of the applications' complexity is as follows:

- Petstore: 3101 LOC, 6 classes, 31 attributes, 30 operations and 5 association relationships.
- Mediaplayer: 2709 LOC, 5 classes, 24 attributes, 26 operations and 6 association relationships.

3.1.4 Implementation environment

The implementation environment had to be able to successfully deal with the kind of important challenges posed by modern Web applications. These challenges include architecture, functionality and user interfaces. To face these new necessities, a number of technologies have been proposed, fostering the adoption of server-side and client-side languages for the Web. Within this plethora of proposals, some technologies have been largely adopted by practitioners.

Among them, *C#* [56] is defined as a simple, modern, object-oriented, and type-safe programming language derived from *C* and *C++*, developed especially for the .NET platform. The .NET runtime components, frameworks, and languages are all tied together under the Visual Studio environment [23]. In our experiment we have

[2] For the sake of simplicity, in this article the term task and the term modification are used as synonims.

chosen *C#* and Visual Studio .NET as representatives of a modern programming environment for the Web that is currently being used by a large community of software practitioners.

With respect to the MDE paradigm, there are two tool mainstreams:

- Some proposals opt to apply the UML profiling mechanism to extend the different UML elements. They define constraints and add tag values that introduce domain-specific semantics. In this category, there is a large set of code generation tools. However, to the best of our knowledge, only two of these tools can generate both an NHibernate-based object-oriented business logic and the persistence layers -a requirement for the two systems used in the experiment-. These are Modelio [4] and Visual Paradigm [2]. Both of them are commercial, and they do not provide academic licenses.
- A second set of proposals, which has been gaining momentum over recent years due to the increasing complexity of Web applications [58], defines Domain-Specific Languages (DSLs). These DSLs require the definition of a metamodel or a grammar to establish the abstract syntax and a graphical or textual language to represent its concrete syntax. Specifically, we will only focus on DSLs of the MDE discipline that recommends define the abstract syntax using metamodels based on the standard MOF. Thus, the DSL developer can freely dispose of a large set of tools that permits him to represent the model and to generate other models using model-to-model or to generate code with model-to-text transformation languages. Among the commercial MDE tools in this second set we can cite Integranova [3] and RadarC [27]. Also, several proposals have been developed in universities, such as OOH4RIA [39], OOHDM [57], RUX [33], WebML [10], UWE [54] and OOWS [59].

Given the fact that none of the above mentioned MDE proposals is standard or even widely adopted, and due to the lack of academic licenses for any of the commercial tools, we have opted to use OOH4RIA as a representative of a modern MDE environment. The OOH4RIA approach extends the OOH method [9] and proposes a complete development process based on a set of models and transformations that allow to go from conceptual models to code. OOH4RIA is also equipped with an Implementation Development Environment (IDE) [40] that offers support for both the design activities and the automatic code generation process. This IDE is based on the Eclipse Modeling Project [21], an open source software whose main purpose is to provide a highly integrated platform tool [8]. OOH4RIA is free of charge for universities, and it is taught as part of the Master's degree in which the experiment took place.

Summarizing, the following implementation environment was set up:

- Development framework: .NET framework, Silverlight 4.0 and NHibernate (Object-Relational Mapping).
- Coding IDE: Visual Studio 2010.
- Modelling tool: OOH4RIA.
- Code Generation tool: OOH4RIA.
- Languages: *C#* and XML Mapping (ORM mapping of NHibernate).

– Other tools: The set of questionnaires filled-in by each developer, which were published online.

3.2 Experiment Planning

As we have mentioned previously, and due the fact that the subjects were not randomly chosen but rather selected on the basis of their attendance to a Master's degree, our study belongs to the quasi-experiment category.

The subjects were randomly assigned to either one of the two available applications, and they were asked to perform ten maintainability tasks, five with the code-centric approach and five with the model-driven approach. Some of the tasks were corrective and some were perfective. For each task type they started with a new project that they downloaded from the replication package [37]. There was no time limit, although the expected time to fulfill the 10 tasks -based on a previous pilot test- was around two hours. The order in which each subject applied each method (.NET and OOH4RIA) was randomized to avoid order effects. Such randomization can be seen in Table 1.

Table 1 Experiment cross-tabulated design

Subject	**Corrections**	**Improvements**	**Application**
S1-S6	OOH4RIA	.NET	Petstore
S7-S13	.NET	OOH4RIA	Petstore
S14-S20	OOH4RIA	.NET	MediaPlayer
S20-S27	.NET	OOH4RIA	MediaPlayer

The subjects were supervised by two instructors in order to control the interaction bias. After each group of tasks, the students were asked to rate their experience regarding the method used.

3.2.1 Variables

Given the research questions presented in Section 3.1, we have defined the following Independent (experimentally manipulated) Variables (IV) or factors:

– Meth: Method, a categorical variable with two levels: .NET and OOH4RIA. It is important to note that, in this experiment, when we refer to method, we are in fact talking about a compound variable (method*tool), due to the coupling of these two variables in our experimental settings.
– App: Application, a categorical variable with two possible values: Petstore and Mediaplayer

We can characterize Meth as a fixed factor (since it includes the two methods we are interested in testing), while App can be regarded as a random factor (since these two applications are just two examples of the application population). The resulting design is a two-way mixed model ANOVA design.

The dependent (measurable) variables (DV) have been defined based on a well-known Method Adoption Model [47], which is rooted in sound evidence from related fields [48], and has been further refined in [38]. This model is presented in Fig. 1. According to this model, the performance of developers is determined by their actual -objective- effectiveness and efficiency while using the method. Satisfaction, in turn, is determined by their perceived usefulness and their perceived ease of use while using the method. The definition of these variables as they are to be understood in the context of our study is as follows:

- Actual Effectiveness (AEffv). It represents the number of tasks both syntactically and semantically correct carried out by the subjects. Semantically correct tasks not only preserve the executability of the program (correct syntax) but they also fulfill the modification requirement. In our study the corresponding variable has been defined as a ratio scale whose value ranges from 0 (the subject did not carry out any task correctly) to 5 (all five tasks were successfully completed with the assigned method).
- Actual Efficiency (AEffc). It represents the tasks carried out per hour. In our study it is defined as a ratio scale. The derived measure associated is calculated as AEffv divided by time.
- Perceived Usefulness (PU). It represents the extent to which a developer believes that using the method will enhance her job performance. In our study, it has been measured through four measures:
  - Perceived Effectiveness (PEffv): A ratio variable, measured through a 5-point questionnaire item, representing the subjective percentage of tasks that the subject thinks she correctly addressed.
  - Perceived Certainty (PCert): An interval variable, measured through a 5-point questionnaire item, representing the subjective certainty of the subject with respect to the correction of the proposed tasks.
  - Perceived Stability (PStab): An interval variable, measured through a 5-point questionnaire item, representing the subjective opinion of the subjects with respect to the proposed tasks not having any collateral or unexpected effect in the application (e.g. the introduction of new errors).
- Perceived Ease of Use (PEU). It represents the degree to which a developer believes that using a particular method will be free of effort. Again, in our study we have divided this concept into:
  - Perceived Efficiency (PEffc): An interval variable, measured through a 7-point questionnaire item, representing the subjective efficiency of the subject while carrying out the maintainability tasks.
  - Perceived Complexity (PCompl): An interval variable, measured through a 7-point questionnaire item, representing the subjective difficulty of carrying out the maintainability tasks.
  - Perceived Learnability (PLearn): An interval variable, measured through a 7-point questionnaire item, representing the subjective opinion of the subjects

with respect to how easy it is to learn how to perform maintainability tasks with the method.

– Perceived Satisfaction (PS): An interval variable, based on a semantic-differential Likert scale made up of 11 7-point items.

This model also establishes the influences among variables. Such influences are represented in Figure 1 as arrows. Particularly, this model states that it is the subjective experience of developers what determines their intention to adopt a given method, and, eventually, their actual behavior (be it adopting or rejecting the method).

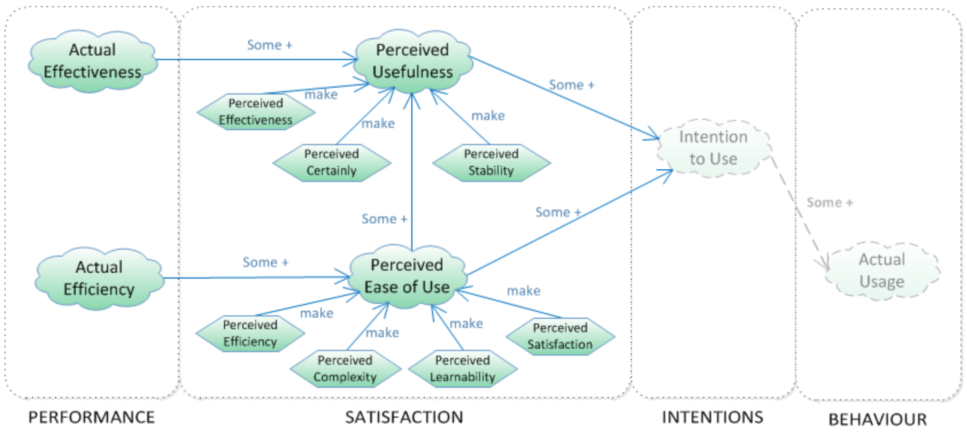


Fig. 1 Theoretical Method Adoption Model components and their associated experimental variables (adapted from [47])

In this figure, the variables (clouds), the measures (hexagons) associated with each variable (arrows labelled *make*) and the positive influences between variables (arrows labeled *some* +) actually put to test by our experiment have been stressed with solid colors and lines. The parts of the model that have been left out of our experiment are marked with shaded colors and dashed lines. In particular, the model states that AEffv positively affects PU, and AEffc positively affects PEU. Additionally, PEU positively affects PU.

3.2.2 Hypotheses

These model dimensions and measures have been used to define the following null and alternative hypotheses, which are based on the research questions presented in Section 3.1.

– Actual Effectiveness Hypothesis (HAEffv, RQ1).
  – $HAEffv_0$: AEffv(OOH4RIA) = AEffv(.NET). The effectiveness of junior software developers while carrying out maintainability tasks with OOH4RIA and with .NET do not significantly differ. This fact holds regardless of the application being developed.

  - $HAEffv_A$: AEffv(OOH4RIA) <> AEffv(.NET).
- Actual Efficiency Hypothesis (HAEffc, RQ1).
  - $HAEffc_0$: AEffc(OOH4RIA) = AEffc(.NET). The efficiency of junior software developers while carrying out maintainability tasks with OOH4RIA and with .NET do not significantly differ. This fact holds regardless of the application being developed.
  - $HAEffc_A$: AEffc(OOH4RIA) <> AEffc(.NET).
- Perceived Efficiency Hypothesis (HPEffc, RQ2).
  - $HPEffc_0$: PEffc(OOH4RIA) = PEffc(.NET). Performing maintainability tasks with the OOH4RIA method and environment make subjects feel as efficient/inefficient as performing maintainability tasks with .NET. This fact holds regardless of the application being developed.
  - $HPEffc_A$: PEffc(OOH4RIA) <> PEffc(.NET).
- Perceived Complexity (HPCompl, RQ2).
  - $HPCompl_0$: PCompl(OOH4RIA) = PCompl(.NET). Subjects feel that performing maintainability tasks with OOH4RIA is as complex as performing them with .NET. This fact holds regardless of the application being developed.
  - $HPCompl_A$: PCompl(OOH4RIA) <> PCompl(.NET)[3].
- Perceived Learnability (HPLearn, RQ2).
  - $HPLearn_0$: PLearn(OOH4RIA) = PLearn(.NET). Subjects feel that learning how to carry out maintainability tasks with OOH4RIA is as easy/difficult as learning to carry them out with .NET. This fact holds regardless of the application being developed.
  - $HPLearn_A$: PLearn(OOH4RIA) <> PLearn(.NET).
- Perceived Satisfaction (HPSatisf, RQ2).
  - $HPSatisf_0$: PSatisf(OOH4RIA) = PSatisf(.NET). Subject feels that using OOH4RIA to carry out maintainability tasks is as satisfying as using .NET. This fact holds regardless of the application being developed.
  - $HPSatisf_A$: PSatisf(OOH4RIA) <> PSatisf(.NET).
- Perceived Ease of Use Hypothesis (HPEU, RQ2).
  - $HPEU_0$: PEU-Mean(OOH4RIA) = PEU-Mean(.NET). Using OOH4RIA to carry out maintainability tasks is regarded by subjects, generally speaking, as easy/difficult as using .NET. This fact holds regardless of the application being developed.
  - $HPEU_A$: PEU-Mean(OOH4RIA) <> PEU-Mean(.NET).
- Perceived Effectiveness Hypothesis (HPEffv, RQ2).
  - $HPEffv_0$: PEffv(OOH4RIA) = PEffv(.NET). Performing maintainability tasks with OOH4RIA make subjects feel as effective/ineffective as performing maintainability tasks with .NET. This fact holds regardless of the application being developed.
  - $HPEffv_A$: PEffv(OOH4RIA) <> PEffv(.NET).
- Perceived Certainty Hypothesis (HPCert, RQ2).

[3] In the analyses this scale item has been reversed so that higher ratings correspond to more positive feelings, that is, to less perceived complexity.

  - $HPCert_0$: PCert(OOH4RIA) = PCert(.NET). Subjects feel as secure/insecure with the result of carrying out maintainability tasks with OOH4RIA as with the result of carrying them out with .NET. This fact holds regardless of the application being developed.
  - $HPCert_A$: PCert(OOH4RIA) <> PCert(.NET).
- Perceived Stability Hypothesis (HPStab, RQ2).
  - $HPStab_0$: PStab(OOH4RIA) = PStab(.NET). Subjects feel that the result of carrying out maintainability tasks with OOH4RIA is as stable as the result of carrying them out with .NET. This fact holds regardless of the application being developed.
  - $HPStab_A$: PStab(OOH4RIA) <> PStab(.NET).
- Perceived Usefulness Hypothesis (HPU, RQ2).
  - HPU_0: PU(OOH4RIA) = PU(.NET). Subjects feel that OOH4RIA is as useful to carry out maintainability tasks as .NET. This fact holds regardless of the application being developed.
  - HPU_A: PU(OOH4RIA) <> PU(.NET).

3.2.3 Experiment instrumentation

The materials used in our quasi-experiment, which are included in a replication package [37], have the following structure:

1. Subject confidential agreement.
2. Subject instruction sheet.
3. Project Booklet. There are two modalities: A (corresponding to the Petstore application) and B (Mediaplayer application). No matter the modality, the contents of the booklet include a) a description of the architecture of the application, b) a functional description of the application and, c) an URL where the subjects can download the application. The downloadable files contain both an OOH4RIA project and a Visual Studio 2010 project.
4. Pre-experiment questionnaire. It includes demographic questions as well as questions about subjects' previous experience with Web application development, Web programming and application modeling.
5. Task sheet. It includes two blocks of tasks: five corrective and five perfective. It also includes a sheet where subjects self-report the time that it has taken them to carry out the tasks in that block.
6. Post-experiment questionnaire. Questions to gather the subjective opinions of the subjects regarding each method: PCert, PEffc, PStab, PEffv, PCompl, PLearn and PSatisf.

To facilitate the comparison, the OOH4RIA and the .NET experiment materials for both applications and both blocks of tasks have been designed to be as similar as possible in terms of both the layout of the information and the information content. Tasks in the two applications were controlled to be equivalent, the only difference being the particular class on which the task was to be performed.

3.3 Operation and data collection procedures

The experiment was carried out during one master session, once the subjects had received all the needed training on both methods. Previous to this session, a pilot test was run with two subjects who had enrolled for the Master the year before. Also prior to the session, the students had filled in the pre-experiment questionnaire. This questionnaire included questions about demographic and previous experience data.

The operation phase of the experiment was defined as follows: Half of the students received Modality A - Petstore - and half of them received Modality B - Mediaplayer -. In both modalities the user had to perform a set of corrective and perfective changeability tasks.

The instructors indicated to each subject which block (corrective or perfective) they were to carry out with .NET, and which one had to be completed with OOH4RIA.

After completing both sets of tasks, the subjects were asked to complete a post-experiment questionnaire, where they had to subjectively rate their efficiency and effectiveness with each method, as well as the perceived certainty, stability, complexity, learnability and satisfaction of their maintainability work.

To maintain the comparability of the data collected during the experiment, no feedback was given to the subjects on their performance with the tasks. We also monitored that no interaction between participants occurred.

The performance-based variables (AEffc and AEffv) were calculated manually by one of the instructors, based on the code and the OOH4RIA projects that the students handed in. All students completed the assignments, so there are no missing values in the gathered data. The time to finish each type of task was self-reported through a form. The subjective variables were recorded with a set of online questionnaires. The tool used to manage such questionnaires is Qualtrics [5]. For the subjective measures, some students did leave some of the questions blank. In such cases, we have followed the strategy of leaving those observations out of the analysis. For this reason, the degrees of freedom of the subjective hypotheses may slightly vary from hypothesis to hypothesis.

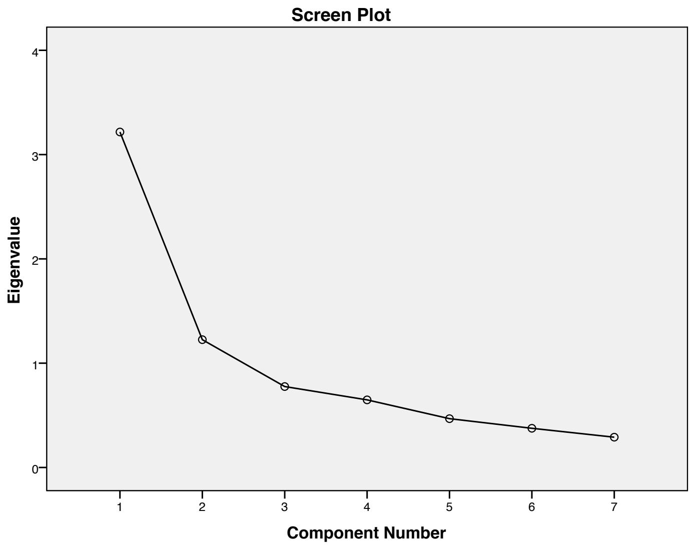
4 Data analysis and interpretation of results

All the analyses have been performed with the PASW (Predictive Analytics SoftWare) package, v18 [52]. The first step of the analysis has consisted in validating the theoretical model on which our hypotheses are based.

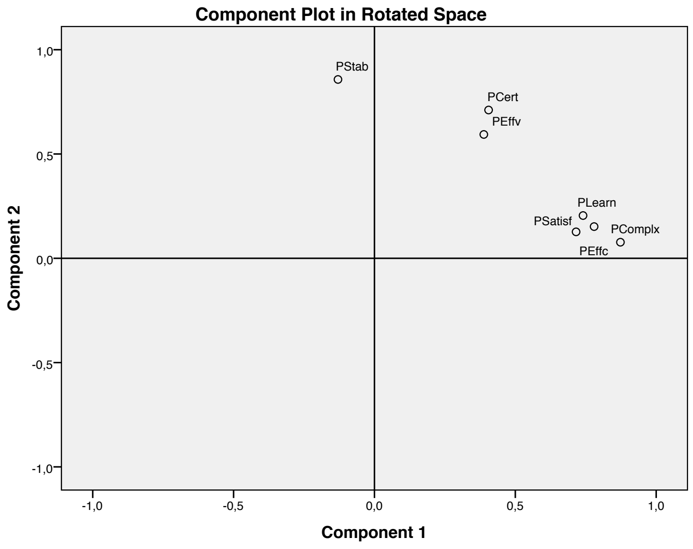
4.1 Theoretical Model Validation

In order to check whether the subjective measures fitted the theoretical model (see Fig. 1), we have performed a Principal Component Analysis (PCA).

The first step of such analysis is to determine the number of components (main variables) underlying the set of measures included in our experiment. The screen plot presented in Fig. 2(a) determines that the optimal number of components to be extracted is two (see the steep slope in Fig. 2(a)).



(a) Screen Plot of initial solution.



(b) The scatterplot matrix.

Fig. 2 PCA plots

Then, the rotated component matrix (varimax rotation) has helped us in determining which measures belong to each component. In the scatterplot matrix of Figure 2(b) we can observe how the first component is made up of four measures (PEffc, PCompl, PLearn, PSatisf), which correspond to the PEU variable in the theoretical model (see Fig. 1), while the second component includes PEffv, PCert and PStab, which are the measures associated with the PU variable in the theoretical model. All of them contribute more than 0.45 to the global construct, which, according to Comrey [14], is the minimum acceptable weight. Therefore we can conclude that our data fits well with the measure grouping suggested by the theoretical model.

4.2 Reliability of the measurement instruments

Also prior to the assessment of the hypotheses, we have checked the reliability of the scales in the context of our experimental settings. We have applied the Cronbach's Alpha test, which has revealed the following results:

- For the PSatisf scale (a semantic-differential scale made up of eleven items), all the items show a correlation higher than 0.3 with the general construct, while the global Cronbach's Alpha is 0.89, giving proof of high internal consistency among the PSatisf items. It is therefore meaningful to calculate the mean, in order to use it as a global rating of PSatisf.
- For the PU scale (made up of three items: PEffv, PCert and PStab, see Fig. 1), although all the items show a correlation higher than 0.3 with the general construct, the Cronbach's Alpha remains rather low ($alpha$ = 0.676), giving proof of low internal consistency among the items. This low internal consistency has prevented us from calculating a global PU value (PU-Mean) by averaging item scorings (see Table 2).
- For the PEU scale (made up of four items: PSatisf, PCompl, PLearn and PEffc, see Fig. 1) the Cronbach's Alpha shows a moderate degree of reliability (α = 0.764), with all items showing an item-total correlation higher than 0.3. This internal consistency makes meaningful to calculate a global PEU value (PEU-Mean) by averaging item scorings (see Table 2).

The remaining subjective measures were single-item scales, so no reliability assessment was possible.

4.3 Descriptive statistics

Table 2 summarizes the means (Mean) and Standard Deviations (SD) for each variable.

Next, we present the analyses of the different hypotheses. Given the number of hypotheses, a summary of the results is presented in Table 3.

Regarding the analyses, it is important to note that, since ANOVA presents a high degree of robustness with respect to both ordinality and non-normality of the scale [51], treating the scales associated with the subjective variables as interval measures does not pose an important threat to the conclusion validity of the study. This is

Table 2 Descriptive statistics (NA: Not Applicable)

Variables	**.NET**		**OOH4RIA**		**Petstore**		**Mplayer**	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
AEffv	3.33	1.11	4.22	0.89	3.92	1.02	3.64	1.16
AEffc	3.84	2.07	12.34	6.21	9.73	7.34	6.56	4.75
PEffv	3.74	0.91	4.23	0.87	4.21	1.03	3.81	0.80
PCert	3.33	0.78	3.74	0.81	3.73	0.72	3.36	0.87
PStab	3.26	0.76	3.07	1.0	3.46	0.58	2.89	1.03
PU-Mean	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*
PEffc	4.57	1.16	5.37	1.21	5.00	1.33	4.96	1.20
PLearn	4.27	1.37	4.62	1.52	4.84	1.49	4.07	1.33
PCompl	3.96	1.40	4.88	1.42	4.68	1.40	4.18	1.52
PSatisf	4.63	0.96	4.56	0.94	4.69	1.00	4.51	0.88
PEU-Mean	4.35	0.88	4.83	1.05	4.77	1.10	4.43	0.86

also consistent with the general opinion that Likert scales that are defined by means of sufficient (typically 7 point) symmetric and equidistant Likert items approximate an interval-level measurement. Furthermore, treating the scale as interval can be beneficial; otherwise, some valuable information (the 'distance' between opinions) could be lost [51].

4.4 RQ1: Performance of treatments

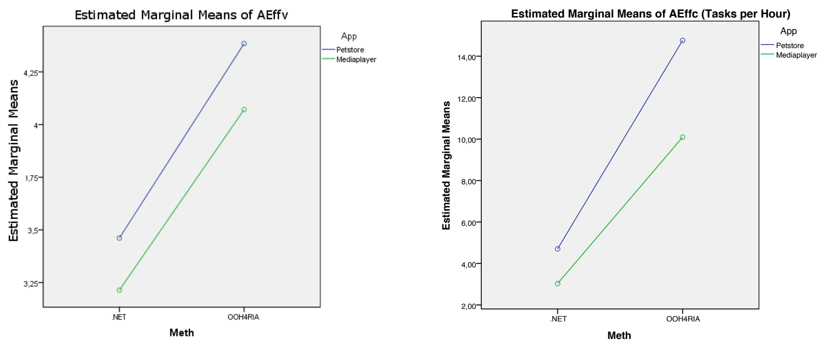
As it was depicted in Figure 1, performance is made up of two variables: actual efficiency and actual effectiveness.

For the refutation of all the hypotheses related to performance we have applied a two-way mixed design ANOVA ($\alpha = 0.05$), in which App is a random factor, Meth is a fixed factor and the different components of performance (AEffv and AEffc) are the DVs. The fact that the number of observations per cell (Meth*App combination) is as equal as possible contributes to the robustness of these analyses.

The results of testing the **HAEffv hypothesis** (concerning the existence of significant differences in the actual effectiveness of the two methods) show that the interaction Meth*App is not significant ($F(1,25) = 0.016, MSE = 0.015, p > 0.05, \eta^2 = 0.001$). We can then safely examine the main effects of the two independent variables (App and Meth) on these means without needing to qualify the results by the existence of a significant interaction. For the Meth variable, the subject's effectiveness when using OOH4RIA ($M = 4.22, SD = 0.89$) is significantly greater than the effectiveness of subjects using .NET code ($M = 3.33, SD = 1.11$): ($F(1,25) = 11.45, MSE = 10.68, p < 0.05, \eta^2 = 0.314$). An effect size (eta squared) of 0.314 means that the factor Meth by itself accounts for 31.4% of the overall (effect+error) variance. The results also show that subject's effectiveness is slightly greater with Petstore ($M = 3.92, SD = 1.02$) than with Mediaplayer ($M = 3.64, SD = 1.16$). However, this difference is not significant ($F(1,50) = 72.25, MSE = 1.06, p > 0.05, \eta^2 =$

0.036). This means that the differences in Effectiveness are significantly affected by the method used, regardless of the particular application being developed.

We can observe these results graphically in Figure 3 (a). The fact that the particular application is not significant is reflected in the lines being quite close to each other. The Meth variable influence is reflected in the slope of the lines. Finally, the lack of significance of the Meth*App interaction is reflected in the lines being more or less parallel (all of them showing the same tendency with each method). The same graphical clues hold for the rest of the graphics.



(a) Actual Effectiveness. (b) Actual Efficiency.

Fig. 3 Objective measures

Then, we have tested the **HAEffc hypothesis** (in tasks per hour).

In this case both the Box's M and the Levene's contrast on the AEffc(.NET) variable were significant. In order to overcome this problem, we have applied a logarithmic transformation of the data: we consider the variable LAEffc now as Ln(AEffc). With the transformed variable, all the ANOVA assumptions hold, which allows us to continue with the analysis.

The results show that the interaction Meth*App is not significant ($F(1,25) = 0.040, MSE = 0.015, p > 0.05$). We can then safely examine the main effects of the two independent variables (App and Meth) on these means without needing to qualify the results by the existence of a significant interaction. For the Meth variable, the subject's efficiency when using OOH4RIA ($M = 12.34, SD = 6.21$) is significantly greater than the subject's efficiency when acting on the .NET code ($M = 3.84, SD = 2.07$): ($F(1,25) = 1197.29, MSE = 18.34, p < 0.05, \eta^2 = 0.658$). The results also show that subject's efficiency is significantly different between Petstore ($M = 9.73, SD = 7.34$) and Mediaplayer ($M = 6.56, SD = 4.75$): ($F(1,25) = 6.21, MSE = 2.167, p < 0.05, \eta^2 = 0.199$). This means that the differences in efficiency are significantly affected both by the method used and by the particular application being developed, although the method has a much bigger impact. We can observe these results graphically in Figure 3 (b).

4.5 RQ2: Satisfaction of treatments

As presented in Figure 1, satisfaction is made up of two main variables: PU and PEU.

On the one hand, PU is made up of three measures: perceived certainty, perceived stability and perceived efficiency. They make up a three-item scale that, however, according to the data gathered in our experiment, has a rather low level of reliability (see Section 4.2). For this reason, we have finally left out of our experiment the HPU hypothesis, since we lack a reliable enough instrument to measure such global construct and we have centered on the partial hypotheses HPEffv, HPCert and HPStab.

PEU, on the other hand, is made up of four different measures: perceived complexity, perceived learnability, perceived satisfaction and perceived efficiency. Since the four items make up a scale with sufficient internal reliability, we can safely assume that all these measures can be used to calculate their mean as the value for the main variable PEU. Therefore, all five hypotheses (HPEffc, HPLearn, HPCompl, HPSatisf and HPEU) can be tested.

Again, for the refutation of all the hypotheses related to both PEU and PU we have applied a two-way mixed design ANOVA ($\alpha = 0.05$), in which App is a random factor, Meth is a fixed factor and the different components of PU are the DVs. The fact that the number of observations per cell (Meth*App combination) is as equal as possible contributes to the robustness of these analyses. Prior to the analyses, we reversed some of the questionnaire items so that higher ranks always corresponded to more positive feelings (more learnable, less complex, more efficient, and so on).

Next, we present the data analyses.

4.5.1 Perceived usefulness

The results of testing the **HPEffv hypothesis** show that the interaction Meth*App is not significant ($F(1,20) = 0.275, MSE = 0.078, p > 0.05, \eta^2 = 0.01$). We can then safely examine the main effects of the two IV (App and Meth) on these means without needing to qualify the results by the existence of a significant interaction. For the Meth variable, the subject's perceived effectiveness when using .NET ($M = 3.77, SD = 0.92$) is slightly lower than the effectiveness of subjects working over OOH4RIA ($M = 4.23, SD = 0.87$). This difference is significant: ($F(1,20) = 8.32, MSE = 2.35, p < 0.05, \eta^2 = 0.29$). The results also show that subject's perceived effectiveness is slightly greater with Petstore ($M = 4.21, SD = 1.03$) than with Mediaplayer ($M = 3.81, SD = 0.80$), although this difference is not significant: ($F(1,20) = 1.83, p > 0.05, \eta^2 = 0.08$). This means that the differences in Perceived Effectiveness are significantly affected by the method used regardless of the application implemented. These results can be graphically observed in Figure 4.

Regarding the **HPCert hypothesis** (concerning the existence of significant differences of subject's certainty with the result of carrying out maintainability tasks with the two methods), the Levene's statistic for the PCert(OOH) variable is significant, which indicates a violation of the statistical method assumptions. In order to overcome this problem, we have applied a logarithmic transformation of the data:

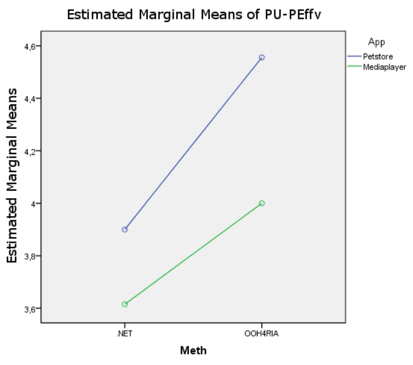
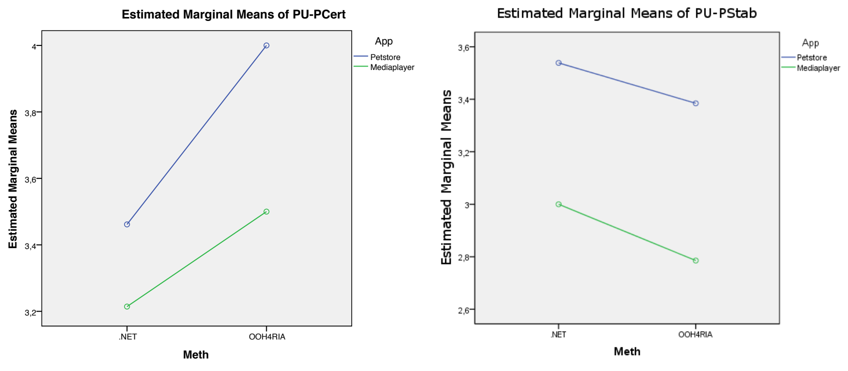


Fig. 4 Perceived Effectiveness

we consider the variable LPCert now as *Ln*(PCert). Again the result of one of the Levene's statistics indicates a violation of homogeneity of error variance. For this reason, another logarithmic transformation has been applied, and now LPCert equals *LnGamma*(PCert). With the transformed variable, we can accept the assumption of homogeneity of variance and continue with the analysis.

The results show that interaction Meth*App is not significant ($F(1, 25) = 0.41, MSE = 0.22, p > 0.05, \eta^2 = 0.01$). Therefore we can safely examine the main effects of the two IV (Meth and App) on these means without needing to qualify the results by the existence of a significant interaction. For the Meth variable, the subject's perceived certainty with respect to the correction of the proposed modifications when using OOH4RIA ($M = 3.74, SD = 0.81$) is slightly higher than the subject's perceived certainty when using .NET ($M = 3.33, SD = 0.78$): ($F(1, 25) = 5.10, MSE = 2.78, p < 0.05, \eta^2 = 0.17$). This main effect is significant. Also, the subject's perceived certainty is slightly higher with Petstore ($M = 3.73, SD = 0.72$) than with Mediaplayer ($M = 3.36, SD = 0.87$): ($F(1, 25) = 2.065, MSE = 1.81, p > 0.05, \eta^2 = 0.07$), although this difference is not significant. This means that the differences in perceived certainty are significantly affected by the method regardless of the application. We can observe these results graphically in Figure 5 (a).

The results of the **HPStab hypothesis** (concerning the existence of significant differences of perceived stability of the subjects with respect to the proposed modifications not having any collateral or unexpected effect in the application with the two methods) show that the interaction Meth*App is not significant ($F(1, 25) = 0.018, MSE = 0.012, p > 0.05, \eta^2 = 0.001$). We can then safely examine the main effects of the two IV (Meth and App) on these means without needing to qualify the results by the existence of a significant interaction. For the Meth variable, the subjects perceived stability of proposed modifications with .NET code ($M = 3.26, SD = 0.76$) is slightly superior to the subjects' perceived stability when using OOH4RIA ($M = 3.07, SD = 1.00$), although not significantly: ($F(1, 25) = 0.67, MSE = 0.457, p > 0.05, \eta^2 = 0.025$). The results also show that the perceived stability of subjects is significantly greater with Petstore ($M = 3.46, SD = 0.58$) than with Mediaplayer ($M = 2.86, SD = 1.03$): ($F(1, 25) = 5.55, MSE = 4.36, p < 0.05, \eta^2 = 0.18$). This



(a) Perceived Certainty. (b) Perceived Stability.

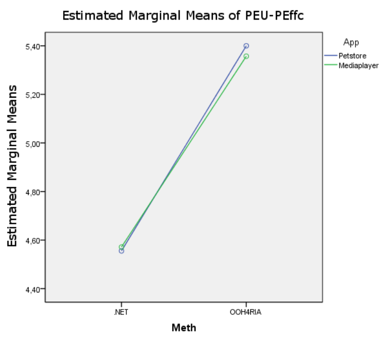
Fig. 5 Certainty and Stability in Perceived Usefulness

means that the differences in perceived stability are significantly affected by the particular application being developed, regardless of the method used. We can observe these results graphically in Figure 5 (b).

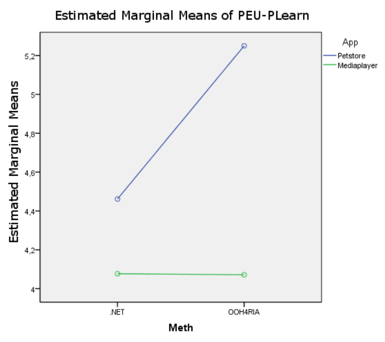
4.5.2 Perceived ease of use

The results of testing the **HPEffc hypothesis** (concerning the existence of significant differences of the subjective performance of the subject while carrying out the maintainability tasks with the two methods) show that the interaction Meth*App is not significant ($F(1,25) = 0.091, MSE = 0.126, p > 0.05, \eta^2 = 0.003$). We can then safely examine the main effects of the two IV (App and Meth) on these means without needing to qualify the results by the existence of a significant interaction. For the Meth variable, the subjects' perceived efficiency when using OOH4RIA ($M = 5.37, SD = 1.21$) is significantly superior to the perceived efficiency of subjects using .NET ($M = 4.57, SD = 1.16$): ($F(1,25) = 6.286, MSE = 8.73, p < 0.05, \eta^2 = 0.23$). Also, subjects' perceived efficiency is slightly higher with Petstore ($M = 5.00, SD = 1.33$) than with Mediaplayer ($M = 4.96, SD = 1.20$): ($F(1,25) = 0.060, MSE = 0.091, p > 0.05, \eta^2 = 0.003$), although not significantly. This means that the differences in perceived efficiency are significantly affected by the method used, regardless of the particular application being developed. The results can be graphically seen in Figure 6 (a).

The results of testing the **HPLearn** hypothesis (concerning the existence of significant differences of subjects' perceived learnability while performing the maintainability tasks with the two methods) show that the interaction Meth*App is not significant ($F(1,23) = 0.916, MSE = 2.133, p > 0.05, \eta^2 = 0.038$). We can then safely examine the main effects of the two IV (Meth and App) on these means without needing to qualify the results by the existence of a significant interaction. For the Meth variable, the subjects' perception of the method learnability with OOH4RIA ($M = 4.62, SD = 1.52$) is slightly better than the perceived learnability with .NET code ($M = 4.27, SD = 1.37$), but this difference is not significant: ($F(1,23) =$



(a) Perceived Efficiency.



(b) Perceived Learnability.

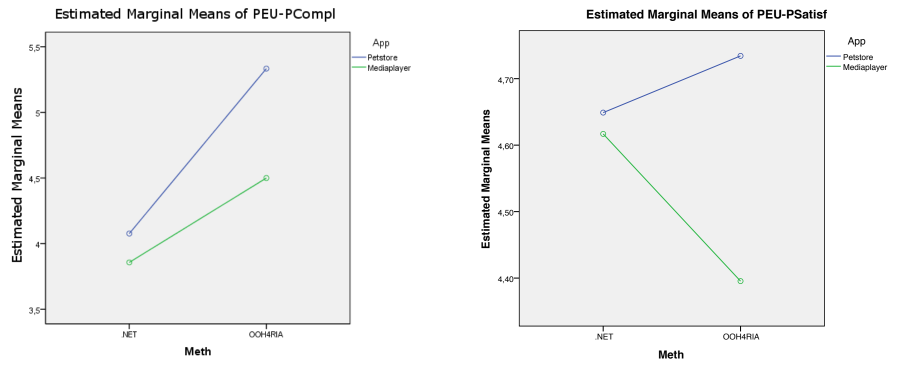
Fig. 6 Efficiency and Learnability in Perceived Ease of Use

$0.607, MSE = 1.413, p > 0.05, \eta^2 = 0.026$). The results also show that the subjects' learnability is significantly greater with Petstore ($M = 4.84, SD = 1.49$) than with Mediaplayer ($M = 4.07, SD = 1.33$): ($F(1,23) = 4.949, MSE = 8.733, p < 0.05, \eta^2 = 0.177$), . This means that differences in perceived learnability are significantly affected by the application, but not by the method. We can observe these results graphically in Figure 6 (b).

The results of testing the **HPCompl hypothesis** (concerning the existence of significant differences of subjects's perceived level of difficulty of carrying out the maintainability tasks with each method) show that the interaction Meth*App is not significant ($F(1,24) = 0.522, MSE = 1.191, p > 0.05, \eta^2 = 0.021$). We can then safely examine the main effects of the two IV (App and Meth). For the Meth variable, the subject's perceived complexity of performing maintainability tasks with OOH4RIA ($M = 4.88, SD = 1.42$) is slightly lower (more positive feelings) than the perceived complexity of performing maintainability tasks over .NET code ($M = 3.96, SD = 1.40$). This difference is significant: ($F(1,24) = 5.076, MSE = 11.576, p > 0.05, \eta^2 = 0.175$). The results also show that the subject's perceived complexity is slightly lower with Petstore ($M = 4.68, SD = 1.41$) than with Mediaplayer ($M = 4.18, SD = 1.52$), although not significantly: ($F(1,24) = 2.069, MSE = 3.627, p > 0.05, \eta^2 = 0.079$). This means that the differences in perceived complexity when subjects carry out maintainability tasks are significantly affected by the method regardless of the application being implemented. These results are graphically depicted in Fig. 7 (a).

To test the **HSatisf hypothesis**, related to the existence of significant differences of subjects's perceived satisfaction while performing the maintainability tasks with the two methods, the Levene's statistic for the PSatisf(OOH) shows a violation of homogeneity of variance. To overcome this problem, we have applied a logarithmic transformation of the data: we consider the variable LPSatisf now as Ln(PSatisf). With the transformed variable, all the assumptions of the test hold, which allows us to continue with the analysis.

The results show that the interaction Meth*App is not significant ($F(1,25) = 0.068, MSE = 0.004, p > 0.05, \eta^2 = 0.003$), which allows us to safely examine



(a) Perceived Complexity. (b) Perceived Satisfaction.

Fig. 7 Complexity and Satisfaction in Perceived Ease of Use

the main effects of the two IV (App and Meth). For the Meth variable, the subjects' perceived satisfaction with OOH4RIA ($M = 4.56, SD = 0.94$) is slightly lower than the perceived satisfaction with .NET code ($M = 4.63, SD = 0.96$), although not significantly: ($F(1,25) = 0.053, MSE = 0.003, p > 0.05, \eta^2 = 0.002$). The results also show that the subjects' perceived satisfaction is slightly superior with Petstore ($M = 4.69, SD = 1.00$) than with Mediaplayer ($M = 4.51, SD = 0.89$), although, again, not significantly: ($F(1,25) = 0.433, MSE = 0.016, p > 0.05, \eta^2 = 0.017$). This means that the differences in perceived satisfaction are not significantly affected neither by the method nor by the application. These results can be graphically seen in Fig. 7 (b).

Last but not least, to test the **HPEU** hypothesis (concerning the existence of significant differences in the global ease of use of the methods while performing maintainability tasks), the Levene's statistic for the PEU(OOH) is significant, indicating a violation of the statistical method's assumption. In order to overcome this problem, we have applied a logarithmic transformation of the data: we consider the variable LPEU now as *Ln*(PEU). Now all the assumptions hold, which allows us to continue with the analysis.

The results of the two-way ANOVA analysis show that the interaction among variables (Meth*App) is not significant ($F(1,25) = 0.106, MSE = 0.005, p > 0.05, \eta^2 = 0.004$). For the Meth variable, subject's perceived global ease of use with OOH4RIA ($M = 4.83, SD = 1.05$) is slightly superior to the perceived ease of use with .NET code ($M = 4.35, SD = 0.88$), although not significantly: ($F(1,25) = 2.696, MSE = 0.14, p > 0.05, , \eta^2 = 0.097$). The results also show the subjects's perceived ease of use is slightly superior with Petstore ($M = 4.77, SD = 1.10$) than with Mediaplayer ($M = 4.43, SD = 0.86$), although, again, not significantly ($F(1,25) = 1.206, MSE = 0.055, p > 0.05, \eta^2 = 0.046$). This means that the differences in perceived global ease of use are not significantly affected neither by the method nor by the particular application being developed. These results can be graphically seen in Figure 8.

A summary of all these results can be seen in Table 3.

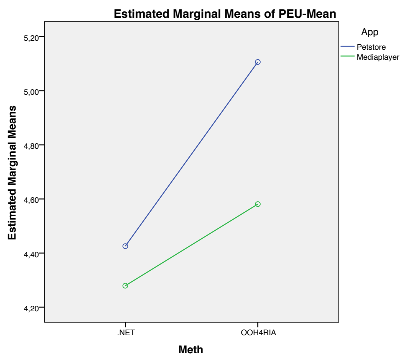


Fig. 8 Perceived Ease of Use

Table 3 Data Analysis Results: Summary (OOH: Significant and better for OOH4RIA, Pet: Significant and better for Petstore, N: Not significant, NA: Not applicable)

Variables	**Meth*App**	**Meth**	**App**	**Meth*App** η^2	**Meth** η^2	**App** η^2
AEffv	N	OOH	N	<0.001	0.314	0.036
AEffc	N	OOH	Pet	<0.001	0.658	0.199
PU-PEffv	N	OOH	N	0.01	0.29	0.084
PU-PCert	N	OOH	N	0.01	0.17	0.076
PU-PStab	N	N	Pet	<0.001	0.025	0.18
PU-Mean	NA	NA	NA	NA	NA	NA
PEU-PEffc	N	OOH	N	0.003	0.23	0.003
PEU-PLearn	N	N	Pet	0.038	0.026	0.177
PEU-PCompl	N	OOH	N	0.021	0.175	0.079
PEU-PSatisf	N	N	N	0.03	0.002	0.017
PEU-Mean	N	N	N	0.004	0.097	0.046

4.6 Threats to Validity

The analysis of the threats to validity evaluates under which conditions our experiment is applicable and offers benefits, and when it might fail [15]. It therefore serves to qualify the experiment results.

Threats to Conclusion Validity refer to issues that affect the ability to draw the correct statistical conclusion about relationships between the treatment and the outcome of the experiment. In our experiment we have performed a sensitivity analysis that assures a power of 0.7 for effect sizes greater than 0.25. The main assumptions of the statistical tests have been checked, and data has been transformed if necessary to fulfill them. The consideration of five and seven-point Likert items as interval variables is a matter of controversy. However, the use of ANOVAs, which present a high degree of robustness with respect to both ordinality and non-normality of the scale [51], significantly reduces this threat. Also,the reliability of measures involving scales has been tested, and hypotheses where the scale did not show a high enough degree of reliability have been left out of the study.

Threats to Internal Validity are concerned with the possibility of hidden factors -not controlled by the researcher- that may be influencing the results. In that case, it would not be possible to conclude that the treatment is the reason for the detected differences in outcome. In our experiment all the subjects enrolled for the master degree had to participate in the experiment (so no selection bias beyond that inherent to quasi-experiments was present). The subjects applied the treatments with different tasks, which diminishes the history risk. Two instructors supervised the whole process in order to diminish the interaction bias. However, students self-reported the time it took them to finish the tasks, which may have introduced a bias. The level of subjectivity involved in the manual correction of tasks by one of the instructors was controlled by executing the code handed in by the subjects, and making sure that it run as expected. The fact that they applied the treatments on two different applications also poses a threat to internal validity that has been controlled by using applications of similar complexity, i.e. similar number of conceptual constructs (for the MDE treatment) and, at the same time, similar number of code lines (for the .NET treatment).

Threats to Construct Validity refer to the generalization of the result to the concept or theory behind the experiment. In this paper we have based the experiment on a widely-accepted theoretical model. A PCA has been performed to check that our data fitted the variable grouping of the theoretical model. This notwithstanding, the fact that some of the variables have been measured through a single questionnaire item may have introduced a measurement bias that can only be overcome with the definition and validation of standard scales to measure such constructs. To our knowledge extent such scales are not yet available for researchers. Also, the hypothesis of the experiment (that is, a higher maintainability of MDE environments) was quite easy to guess, so students may have felt bound to report less time when using MDE. Anyway, the experiment observers took special care not to disclose this hypothesis to the students. Additionally, the experiment suffers from a restricted generalizability across constructs: we have checked a positive outcome between maintainability and OOH4RIA, but we cannot assure that this does not hamper other quality attributes or any additional characteristic.

Last but not least, **Threats to External Validity** are concerned with generalization of the results to the industrial practice. The subjects are graduate students (M.Sc. students), many of them already working in industry as developers, and therefore representatives of junior developers. However, the sample may not be representative of the population of junior developers due to the small sample used and the fact that these subjects are highly motivated to learn new skills. Also, the particular methods and languages we have used (OOH4RIA, .NET), despite being some of them broadly used in industry, constitute a limited environment, and results cannot be generalized to other languages, methods and tools. The applications chosen, although limited in size, are representatives of the kind of applications that are being developed in industry, and were in fact simplified versions of two real projects. However, both the systems and the scope of the maintainability tasks had to be simplified due to time constraints, so there is a risk that different results may happen if bigger applications or different maintainability tasks had been used to perform the experiment. Therefore, this experiment needs to be replicated with different languages, tools, applications, application sizes, tasks and MDE approaches. For this purpose, the replication package of this experiment can be found at [37].

5 Related work

The importance and impact of maintainability on software development is reflected in the vast amount of empirical work found in literature. Regarding code-centric approaches, there are empirical studies which provide evidences about the impact of the quality of object-oriented implementation over the software maintainability. [16] demonstrates that object-oriented metrics such as size, coupling, cohesion, inheritance, etc. can effectively be used to predict the maintainability of software systems. The results indicate that size and direct/import coupling metrics are significant predictors for measuring maintainability of classes while inheritance, cohesion and indirect/export coupling measures are not.

Other studies are focused on how software development problems and good practices may affect software maintainability. On the one hand, [12] classifies a set of problem factors related with software development that may affect the maintainability. These include documentation quality problems, programming quality problems, system requirements problems, personnel resources problems and process management problems. Results suggest that improvements in the software development process can help to reduce the level of severity of the problems related with documentation quality and process management, which in turn may enhance software maintainability. On the other side, [7] estimates the impact of the development activities with a more practical perspective on the software maintainability. This research indicates that the software complexity is a key intermediate variable that links design and development decisions to their downstream effects on software maintenance. Interestingly, the experiment results indicate that the use of an MDE method in development is associated with increased software complexity and reduced software development effort. These results suggest an important link between software development practices and maintenance performance.

Also, according to a systematic mapping of the empirical evidence of the impact of MDE on the different ISO quality characteristics [36], the empirical assessment of maintainability activities while using MDE methods has been gaining momentum during the last years. However, the empirical evidence regarding this topic is still scarce: out of the 599 publications included in the study, only 14 (that is, barely a 2.34%) focused on maintainability. The available evidence states that, in academic settings, it takes around 37% less time to evaluate the impact of a change with an MDE method than directly over the code [41]. In industry, evidence has been gathered on how the use of MDE causes defect reductions, reduced need for code inspections, and produces maintenance gains, although such evidence is anecdotal [46]. In [26], based on the results of an online survey, the authors report how the majority of respondents (between 58 and 66%) considered the use of MDE on their projects to be beneficial in terms of personal and team productivity, maintainability and portability. Organizationally, the benefits of MDE have been defined in terms of communication and control: according to the same survey, almost 75% of respondents agreed that using MDE made them faster at implementing new requirements, while nearly two thirds of respondents reported that their use of MDE helped in the understanding and communication of organizational knowledge among stakeholders.

Last but not least, a recent empirical study [35] compared an MDE method, WebML [10], and a code-centric method (based on PHP), with respect to the performance and satisfaction of junior software developers while executing analyzability, corrective and perfective maintainability tasks on Web applications. The experiment presented in this paper is a follow-up of the WebML empirical study. Both studies provide interesting results on the impact of MDE approaches on the maintainability of Web applications. Also, in the experiment presented in this paper we have maintained the paradigms (model-driven vs code-centric). However, we have changed the methods (now OOH4RIA and .NET). Also, in this study we have centered on changeability -leaving out analyzability-, and we have designed more tasks that provide a finer-grained measure for the maintainability sub-characteristic. Besides, we have measured much more extensively the developer's perceptions, based on an underlying theoretical model. Last but not least, whereas the WebML experiment is mainly focused on the use of the navigational model for obtaining the client-side of a Web application, this work is centered on the definition of the domain model to generate the server-side, that is, the business logic and the persistence layer of a Web application. Comparing the results regarding the productivity of the changeability and the satisfaction of each MDE approach versus the code-centric approach, we can observe how both WebML and OOH4RIA significantly improve the actual efficiency of the changeability tasks (3.21 and 317 times, respectively) with respect to the code-centric approach. However, while OOH4RIA also improves the effectiveness of the changeability by up to 27%, the impact of WebML over effectiveness was not found significant. With regard to the results on the satisfaction of use of these approaches, the WebML study showed a slight preference of subjects towards tackling maintainability tasks directly over the source code. This result is consistent with the results of the OOH4RIA experiment.

6 Conclusions

This study compares a model-driven approach, OOH4RIA, and a code-driven approach, based on Visual Studio and .NET. The study concludes that the use of OOH4RIA -as compared to .NET- greatly improves the actual performance (AEffv and, even more, AEffc) of subjects carrying out maintainability tasks. The study of the satisfaction-related variables (PEU and PU), however, throws mixed results: while PU-PEffv, PU-PCert, PEU-PEffc and PEU-PCompl do reveal significant differences among methods (all in favor of OOH4RIA), PU-PStab, PEU-PLearn, PEU-PSatisf and global PEU don't reveal significant differences between the two methods.

The significant differences all agree with previous results (see section 5). These results already suggested that, in an experimental setting such as ours, the MDE approach would outrank the code-centric approach both in terms of efficiency (actual and perceived), effectiveness (actual and perceived). Also, in these four variables the

Meth variable shows the greater effect size. The significant differences for PCert are also consistent with our a-priori thoughts. Models provide higher-level abstractions that are assumed to improve the understandability of the domain, including the understandability of the maintainability changes.

However, we also expected significant differences in the variables that actually did not attain significance. Users found OOH4RIA solution to be slightly less stable (meaning that they fear that more unnoticed errors may have been introduced), although the difference was not significant. This happens even though OOH4RIA includes a checking mechanism that is aimed at helping developers to avoid the introduction of unnoticed errors (a mechanism that was not available for the .NET treatment). We believe that this may be due to the perceived loss of control with MDE approaches that subjects have reported in a related study [38]. Also, subjects believe that OOH4RIA is slightly more learnable and less complex than .NET (although, again, not significantly). We expected that, having all of them a much higher experience with .NET than with OOH4RIA, they would find .NET to be more learnable and less complex. We can partly explain this fact by the subjects' knowledge of UML, which apparently was strong enough so as not to feel discomfort with the OOH4RIA approach. Also, the use of models -whose higher level of abstraction is supposed to simplify the maintenance tasks- seems to counterbalance the additional complexity that is likely to be perceived when the developer is faced with a new method. Also, the fact that the subjects were junior developers attending a master where they were being exposed to many different tools and techniques, may have had an impact on such perception of complexity. Last but not least, the subjects reported a PSatisf level slightly lower for OOH4RIA than for .NET (although, again, not significantly). We expected subjects to be significantly more satisfied with OOH4RIA, given the expected gains in efficiency and effectiveness. Again, based on self-reported opinions disclosed in a previous experiment [38], we believe that this lower level of satisfaction (despite the increased objective and subjective productivity gains) may be due to code-centric practices being more consistent with the existing values, needs and past experiences of subjects, which suggests the convenience to introduce this variable in future replications of the experiment.

Since it is well known that it is subjective opinions (and not objective behavior) what better determines the intention to adopt new technologies and methods [49], it is not far-fetched to think that the discrete satisfaction results reflected in our data (with 4 variables being significant and 3 variables not being significant) may be at the root of the low adoption rates of the MDE paradigm beyond university courses [60]. Actually checking whether this hypothesis is true requires from further experimentation.

Another important contribution of this article is the validation, through a PCA, of the structure of the theoretical model. Our analysis shows that the number of components and the assignment of measures to model variables is consistent with the data gathered in our context, and therefore increases the confidence in the goodness of fit of the chosen theoretical model to the context of method adoption.

Our results augment the repository of empirical data comparing maintainability performance and satisfaction of MDE methods with respect to traditional code-centric approaches. Further experimentation is needed to be able to generalize the results to a different population, different methods and languages, different application types, different maintainability tasks or different application sizes. In this sense, we have prepared a replica, and we plan to run a meta-analysis that includes the results from the WebML experiment, the experiment reported in this paper and the results from that replica. Also, extensive work needs to be done in the matter of defining reliable measures. Last but not least, more data is needed to perform a much more extensive validation of the theoretical model. In this paper we have just checked the soundness of the groupings of the measures under the PEU and PU variables. However, we have left out some constructs of interest (namely, the intention to use the method and how it relates to the actual use of the method once the subjects go back to their daily activities), and we have not checked the causal relationships inferred from that model.

Acknowledgements This paper has been co-supported by the DLSI, the Spanish Ministry of Education, and the University of Alicante under contracts TIN2010-15789 (SONRIA) and GRE10-23 (DISEMRIA). The authors wish to thank their students for taking the time to participate in this empirical study. In addition, we would also like to thank Jose Javier Martínez and Juan Antonio Osuna who contributed to the development of the OOH4RIA Tool. Special thanks to Dania Suárez and Milton García for their help.

References

1. ISO/IEC 9126-1, Software engineering - Product quality - Part 1: Quality model (2001)
2. Visual Paradigm for UML 8.0 Community Edition. http://www.visual-paradigm.com/ (Last update August 16, 2010) (2005)
3. Integranova M.E.S (Integranova Software Solutions). http://www.slideshare.net/mhoubraken/catalogo-integranova-esp-mayo-2011 (2011)
4. Modelio open source modeling environment v2.2. http://www.modelio.org (2011)
5. Qualtrics online survey software. http://www.qualtrics.com/ (2012)
6. Ameller, D., Franch Gutiérrez, J., Cabot Sagrera, J.: Dealing with non-functional requirements in model-driven development. Tech. rep., Departament d'Enginyeria de Serveis i Sistemes d'Informació (2010)
7. Banker, R.D., Davis, G.B., Slaughter, S.A.: Software development practices, software complexity, and software maintenance performance: A field study. Management Science **44**(4), 433–450 (1998)
8. Budinsky, F.: Eclipse modeling framework: a developer's guide. Prentice Hall Ptr (2004)
9. Cachero, C., Poels, G., Calero, C.: Towards a Quality-Aware Web Engineering Process. In: Proceedings of the 12th International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD'2007), pp. 7–16. Citeseer (2007)
10. Ceri, S., Fraternali, P., Matera, M.: Conceptual modeling of data-intensive Web applications. Internet Computing, IEEE **6**(4), 20–30 (2002)
11. Chapin, N., Hale, J.E., Khan, K.M., Ramil, J.F., Tan, W.G.: Types of software evolution and software maintenance. Journal of Software Maintenance and Evolution: Research and Practice **13**(1), 3–30 (2001)
12. Chen, J.C., Huang, S.J.: An empirical analysis of the impact of software development problem factors on software maintainability. Journal of Systems and Software **82**(6), 981–992 (2009)
13. Cohen, J.: Statistical power analysis for the behavioral sciences. Lawrence Erlbaum (1988)
14. Comrey, A.L.: A First Course in Factor Analysis. LawreAcademic Pressnce Erlbaum (1973)
15. Cook, T.D., Campbell, D.T., Day, A.: Quasi-experimentation: Design & analysis issues for field settings. Houghton Mifflin Boston (1979)
16. Dagpinar, M., Jahnke, J.H.: Predicting maintainability with object-oriented metrics-an empirical comparison. In: Proceedings of the 10th Working Conference on Reverse Engineering (WCRE), pp. 155–164 (2003)
17. Dyba, T., Kitchenham, B.A., Jorgensen, M.: Evidence-based software engineering for practitioners. Software, IEEE **22**(1), 58–65 (2005)

18. Faul, F., Erdfelder, E., Lang, A.G., Buchner, A.: G*power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences. Behavior Research Methods **39**, 175–191 (2007)
19. Fu, X., Shi, W., Akkerman, A., Karamcheti, V.: Cans: composable, adaptive network services infrastructure. In: Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems-Volume 3, pp. 12–12. USENIX Association (2001)
20. Glitho, R.H., Khendek, F., De Marco, A.: Creating value added services in internet telephony: an overview and a case study on a high-level service creation environment. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on **33**(4), 446–457 (2003)
21. Gronback, R.C.: Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit. Addison-Wesley Professional (2009)
22. Gustavsson, H., Lings, B., Lundell, B., Mattsson, A., Beekveld, M.: Integrating proprietary and open-source tool chains through horizontal interchange of XMI models. In: Software Maintenance, 2007. ICSM 2007. IEEE International Conference on, pp. 521–522. IEEE (2007)
23. Hanselman, S., Verma, D.: Visual Studio 2010 SP1 for Web Developers. MSDN Magazine-Louisville (2011)
24. Heijstek, W., Chaudron, M.R.V.: Empirical investigations of model size, complexity and effort in a large scale, distributed model driven development process. In: Software Engineering and Advanced Applications, 2009. SEAA'09. 35th Euromicro Conference on, pp. 113–120. IEEE (2009)
25. Hutchinson, J., Whittle, J., Rouncefield, M.: Empirical assessment of mde in industry. Interpretation A Journal Of Bible And Theology pp. 471–480 (2011). <m:note/>
26. Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.: Empirical assessment of MDE in industry. In: Proceeding of the 33rd International Conference on Software Engineering, pp. 471–480. ACM, ACM (2011)
27. Icinetic TIC S.L.: RADARC (Rapid Application Development Architecture). http://www.radarc.net/ (2012)
28. IEEE Std.1219-1998: IEEE Standard for Software Maintance (1998)
29. ISO/IEC FCD 25010: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models (2011)
30. Kampenes, V., Dyba, T., Hannay, J., Ksjoberg, D.: A systematic review of quasi-experiments in software engineering. Softw. Technol. **51**, 71–82 (2009)
31. Kitchenham, B., Al-Khilidar, H., Babar, M., Berry, M., Cox, K., Keung, J., Kurniawati, F., Staples, M., Zhang, H., Zhu, L.: Evaluating guidelines for reporting empirical software engineering studies. Empirical Software Engineering **13**(1), 97–121 (2008)
32. Kitchenham, B., Budgen, D., Brereton, P., Turner, M., Charters, S., Linkman, S.: Large-scale software engineering questions-expert opinion or empirical evidence? Software, IET **1**(5), 161–171 (2007)
33. Linaje, M., Preciado, J., Sánchez-Figueroa, F.: A method for model based design of rich internet application interactive user interfaces. Web Engineering pp. 226–241 (2007)
34. López, E., González, M., López, M., Iduñate, E.: Proceso de Desarrollo de Software Mediante Herramientas MDA. Revista Iberoamericana de Sistemas, Cibernética e Informática **3**(2), 6–10 (2006)
35. Martinez, Y., Cachero, C., Matera, M., Abrahao, S., Luján, S.: Impact of MDE Approaches on the Maintainability of Web Applications: An Experimental Evaluation. In: Conceptual Modeling-Er 2011: 30th International Conference on Conceptual Modeling, Brussels, Belgium, October 31-November 3, 2011. Proceedings, vol. 6998, pp. 233–246. Springer-Verlag New York Inc, Springer-Verlag New York Inc (2011)
36. Martinez, Y., Cachero, C., Meliá, S.: Evidencia empírica sobre mejoras en productividad y calidad mediante el uso de aproximaciones MDD: un mapeo sistemático de la literatura. (2011)
37. Martnnez, Y., Cachero, C., Meliá, S.: Experiment replication package. http://artemisa.dlsi.ua.es/ooh4ria/labPackages/Maintainability2012.rar (2012)
38. Martnnez, Y., Cachero, C., Meliá, S.: Mdd vs. traditional software deveopment: A practitioner's subjective perspective. Softw. Technol. (2012). URL `http://dx.doi.org/10.1016/j.infsoft.2012.07.004`
39. Meliá, S., Gómez, J., Pérez, S., Díaz, O.: A model-driven development for GWT-based Rich Internet Applications with OOH4RIA. In: Web Engineering, 2008. ICWE'08. Eighth International Conference on, pp. 13–23. IEEE (2008)
40. Meliá, S., Martínez, J.J., Mira, S., Osuna, J., Gómez, J.: An Eclipse Plug-in for Model-Driven Development of Rich Internet Applications. Web Engineering pp. 514–517 (2010)

41. Mellegå rd, N., Staron, M.: Improving Efficiency of Change Impact Assessment Using Graphical Requirement Specifications: An Experiment. Product-Focused Software Process Improvement pp. 336–350 (2010)
42. Mellor, S.J., Clark, T., Futagami, T.: Model-driven development: guest editors' introduction. IEEE software **20**(5), 14–18 (2003)
43. Mens, T., Guéhéneuc, Y., Fernández-Ramil, J., D'Hondt, M.: Guest editors' introduction: software evolution. Software, IEEE **27**(4), 22–25 (2010)
44. Mens, T., Van Gorp, P.: A taxonomy of model transformation. Electronic Notes in Theoretical Computer Science **152**, 125–142 (2006)
45. Mohagheghi, P., Conradi, R.: An empirical study of software change: origin, acceptance rate, and functionality vs. quality attributes. In: Empirical Software Engineering, 2004. ISESE'04. Proceedings. 2004 International Symposium on, pp. 7–16. IEEE (2004)
46. Mohagheghi, P., Dehlen, V.: Where is the proof? - A review of experiences from applying MDE in industry. In: European Conference on Model Driven Architecture–Foundations and Applications (ECMDA 2008), pp. 432–443. Springer (2008)
47. Moody, D.: The method evaluation model: a theoretical model for validating information systems design methods. In: Proceedings of the 11th European Conference on Information Systems, pp. 16–21. Naples, Italy, June (2003)
48. Moody, D.L.: Dealing with Complexity: A Practical Method for Representing Large Entity Relationship Models (PhD Thesis). Melbourne, Australia: Department Of Information Systems, University of Melbourne (2001)
49. Moore, G.C., Benbasat, I.: Development of an instrument to measure the perceptions of adopting an information technology innovation. Information systems research **2**(3), 192–222 (1991)
50. Muñoz, J., Pelechano, V.: MDA vs Factorías de Software. Actas del II Taller sobre Desarrollo de Software Dirigido por Modelos, MDA y Aplicaciones (DSDM 2005) p. 1 (2005)
51. Norman, G.: Likert scales, levels of measurement and the laws of statistics. Advances in health sciences education **15**(5), 625–632 (2010)
52. Norusis, M.J., et al.: PASW Statistics 18 guide to data analysis. Prentice Hall Press (2010)
53. Perry, D.E., Porter, A.A., Votta, L.G.: Empirical studies of software engineering: a roadmap. In: Proceedings of the conference on The future of Software engineering, pp. 345–355. ACM (2000)
54. Preciado, J.C., Linaje, M., Morales-Chaparro, R., Sanchez-Figueroa, F., Zhang, G., Kroiß, C., Koch, N.: Designing rich internet applications combining uwe and rux-method. In: Web Engineering, 2008. ICWE'08. Eighth International Conference on, pp. 148–154. IEEE (2008)
55. SUN Microsystems: Java Pet Store Sample Application, Blueprints Online (2010). URL `http://java.sun.com/reference/blueprints/index.html`
56. Turtschi, A.: C#. NET Web developer's guide. Syngress Media Incorporated (2002)
57. Urbieta, M., Rossi, G., Ginzburg, J., Schwabe, D.: Designing the interface of rich internet applications. In: Web Conference, 2007. LA-WEB 2007. Latin American, pp. 144–153. IEEE (2007)
58. Vallecillo, A., Koch, N., Cachero, C., Comai, S., Fraternali, P., Garrigós, I., Gómez, J., Kappel, G., Knapp, A., Matera, M., Others: MDWEnet: A practical approach to achieving interoperability of model-driven Web engineering methods. In: Workshop Proc. of 7th Int. Conf. on Web Engineering (ICWE'07), Italy. Citeseer (2007)
59. Valverde, F., Pastor, O.: Facing the Technological Challenges of Web 2.0: A RIA Model-Driven Engineering Approach. Web Information Systems Engineering-WISE 2009 pp. 131–144 (2009)
60. Walderhaug, S., Mikalsen, M., Benc, I., Erlend, S.: Factors affecting developers' use of MDSD in the Healthcare Domain: Evaluation from the MPOWER Project. In: From Code Centric to Model Centric Software Engineering: Practices, Implications and ROI. Workshop at European Conference on Model-Driven Architecture (2008)
61. Wohlin, C., Runeson, P., Höst, M.: Experimentation in software engineering: an introduction. Springer Netherlands (2000)
62. Zelkowitz, M.V.: An update to experimental models for validating computer technology. Journal of Systems and Software **82**(3), 373–376 (2009)

Appendix A: Post-experiment questionnaire

Perceived Usefulness

Please, rate your degree of agreement with the following sentences while performing the maintainability tasks:

	1	2	3	4	5
PU-PEffv: Using OOH4RIA increases the number of maintainability tasks that I can correctly accomplish					
PU-PCert: Using OOH4RIA increases my certainty about the correction of the modifications that I perform					
PU-PStab: Using OOH4RIA reduces the collateral or unexpected effects of the modifications that I perform					
PU-PEffv: Using .NET increases the number of maintainability tasks that I can correctly accomplish					
PU-PCert: Using .NET increases my certainty about the correction of the modifications that I perform					
PU-PStab: Using .NET reduces the collateral or unexpected effects of the modifications that I perform					

Perceived Ease of Use

Please, rate your degree of agreement with the following sentences while performing the maintainability tasks:

	1	2	3	4	5	6	7
PEU-PEffc: Using OOH4RIA enables me to accomplish maintainability tasks more quickly							
PEU-PCompl: Using OOH4RIA makes it easier for me to carry out a maintainability task							
PEU-PLearn: Using OOH4RIA requires me a lot of training to perform a maintainability task							
PEU-PEffc: Using .NET enables me to accomplish maintainability tasks more quickly							
PEU-PCompl: Using .NET makes it easier for me to carry out a maintainability task							
PEU-PLearn: Using .NET requires me a lot of training to perform a maintainability task							

In general, I feel that OOH4RIA is...

PEU-PSatisf	**1**	**2**	**3**	**4**	**5**	**6**	**7**
(1) Difficult to use .. Easy to use (7)							
(1) Tiring .. Not tiring (7)							
(1) Stranger .. Family(7)							
(1) Useless .. Useful (7)							
(1) Demanding .. Simple (7)							
(1) Inefficient .. Efficient (7)							
(1) Boring .. Fun (7)							
(1) Unreliable .. Reliable (7)							
(1) Stressful .. Relaxing (7)							
(1) Unpleasant to use .. Pleasant to use (7)							
(1) Unacceptable .. Acceptable (7)							

In general, I feel that .NET is...

PEU-PSatisf	**1**	**2**	**3**	**4**	**5**	**6**	**7**
(1) Difficult to use .. Easy to use (7)							
(1) Tiring .. Not tiring (7)							
(1) Stranger .. Family(7)							
(1) Useless .. Useful (7)							
(1) Demanding .. Simple (7)							
(1) Inefficient .. Efficient (7)							
(1) Boring .. Fun (7)							
(1) Unreliable .. Reliable (7)							
(1) Stressful .. Relaxing (7)							
(1) Unpleasant to use .. Pleasant to use (7)							
(1) Unacceptable .. Acceptable (7)							