A Family of Experiments to Compare Two Model-Driven Development Tools vs a Traditional Development Method

Jose Ignacio Panach, Óscar Pastor, Natalia Juristo

Abstract— Context: There are many papers which extol the benefits of Model-Driven Development (MDD) compared to traditional developments. However, the adoption of MDD to develop fully functional systems without coding is not frequent. **Objective:** This paper presents a family of experiments with 7 replications and 56 sample units to compare a traditional method versus MDD, analysing two MDD tools. **Method:** The factor in the experiment is the method with two treatments (traditional and MDD). We analyse together all replications thanks to two moderator variables in a multilevel hierarchy: Replication (from 1 to 7) and MDD Tool (INTEGRANOVA and WebRatio). Response variables are Functional Suitability, measured in terms of effectiveness in simple and complex problems; Effort, measured as time to develop simple problems; and Satisfaction, measured in terms of perceived ease of use, perceived usefulness and intention to use. **Result:** Functional Suitability in MDD yields significantly better results both in simple and complex problems, while Effort for simple problems is also significantly better for MDD. Differences for Functional Suitability in complex problems are greater than with simple ones. Considering the MDD tools, INTEGRANOVA yields significantly better Functional Suitability in complex problems. Regarding Satisfaction, replications with WebRatio have a better intention to use than replications with INTEGRANOVA. **Conclusions:** Even though MDD yields better Functional Suitability and Effort, the subjects' Satisfaction with MDD is not better than with a traditional method.

Index Terms— D.1.2 Automatic Programming; D.2.1.e Methodologies; D.2.1.i Validation

1. INTRODUCTION

Model-Driven Development (MDD) is a term used to describe the systematic use of software abstractions (models) as primary artifacts during a software engineering process [1]. Even though the idea of using MDD does not involve the use of automatic code generation from models, the MDD community has historically tried to automate the process as much as possible. This way developers' effort can focus on building conceptual models which abstractly represent the system, and then model-to-code transformations implement the code [2]. There are numerous ideas that appear under the umbrella of MDD approaches [3]: (1) Metaprogramming: where a model creates a programming template; (2) Domain-specific language: a computer language specialized to a particular application domain; (3) Domain-specific modelling: a modelling language is customized to a particular domain; (4) Generative programming: which models families of software systems so that code can be automatically generated from these families; (5) Model-integrated computing, which aims to integrate heterogeneous systems of different domains; (6) Generic model management: which addresses techniques for threatening models and mappings; (7) Conceptual model programming: where analysts' effort focus on building holistic conceptual models and all code is automatically generated. From all these approaches of MDD, the contribution of this article focuses on the last one. From now on, when we refer to MDD, we only refer to the perspective of **conceptual model programming**.

In contrast to MDD, we have the traditional software development method where the developer has to manually write the code that implements the system. The use of models in this case is only for documentation, or as instruments to report solutions, with no option to generate code from them. This leads to both poor and obsolete models which do not correspond with the implementation.

There are works which extol the benefits of MDD, such as the work of Hailpern and Tarr [4], Selic [5] or Weigert and Weil [6]. However, from an empirical point of view, there are not many families of experiments that have contrasted some of the advantages of MDD. We have carried out two previous works on this research line. In [7] we conducted an experiment with 13 sample units to compare MDD versus a traditional method in terms of Functional Suitability (how well the software complies with the requirements), Effort (spent time in simple problems), Productivity (Functional Suitability/Effort), and Satisfaction (pleasant feeling). The results did not yield any significant difference even though we noted, through descriptive data, a trend that shows that MDD results in more Functional Suitability for complex problems. In order to

- *J.I. Panach is a faculty member of the Escola Tècnica Superior d'Enginyeria, Departament d'Informàtica, Universitat de València, Avinguda de la Universitat, s/n 46100 Burjassot, València, Spain. E-mail: joigpana@uv.es*
- *N. Juristo, is a faculty member of the Escuela Técnica Superior de Ingenieros Informáticos, Universidad Politécnica de Madrid, Campus de Montegancedo, 28660 Boadilla del Monte, E-mail: natalia@fi.upm.es*
- *O. Pastor is a faculty member of the Centro de Investigación en Métodos de Producción de Software,Universitat Politècnica de València, Camino de Vera s/n, Edificio 1F, 46022 Valencia, Spain. E-mail: opastor@pros.upv.es*

improve the statistical power and investigate this line, in [8] we conducted six replications of the same experiment in two different universities. We gathered 52 sample units, with enough power to avoid type II errors; the failure to reject a false null hypothesis as the result of a test procedure. We concluded that MDD yields better Functional Suitability, independently of problem complexity.

Starting from the conclusions extracted from both previous works ([7] and [9]), the main contribution of this article is the comparison of MDD versus a traditional method, independently of the MDD tool used in the development process. The MDD paradigm is strongly associated with the MDD tool that operationalized this process, and we would like to study if the results can be generalized for different MDD tools. The context being analysed is the development of a fully functional system from scratch. Therefore, we conducted a family of 7 experiments with two MDD tools: 4 replications with the MDD tool INTEGRANOVA [10] and 3 replications with WebRatio [11]. 4 out 3 replications with the MDD tool INTEGRANOVA are based on the raw data published at [9]. Note that in this article we have gathered all replications (previously published and new ones) except for the baseline, which we have not considered since their problems were not the same as the ones used in the rest of the replications. The few sample units of the baseline (13), together with the complexity to aggregate data with different problems, lead us to discard using the baseline in our current analysis. The aggregation of all replications results in 56 experimental units, which provide sufficient statistical power (at least 0.8) to avoid type II errors with 0.05 probability. This family of experiments is based on one factor named Method (with two treatments: Traditional and MDD) and two moderator variables to aggregate the replications in a multilevel hierarchy: **Replication** (from 1 to 7) and **MDD Tool** (INTEGRANOVA and WebRatio). As response variables, we have Functional Suitability (measured in terms of effectiveness in simple problems and effectiveness in complex problems), Effort (time spent for simple problems) and Satisfaction (measured in terms of Perceived Ease of Use, Perceived Usefulness and Intention to Use). These response variables are the same ones we used in our first experiment [7]. Productivity is not analysed because it is derived from Effort/Functional Suitability [12]; and both Effort and Functional Suitability are considered in our family.

The article is organized as follows. **Section 2** discusses related work. **Section 3** describes how a traditional method and MDD are operationalized in a software development process. **Section 4** presents the characteristics shared by all replications and the differences among them. **Section 5** presents the outcomes. **Section 6** discusses the interpretation of the results. Finally, **Section 7** shows the conclusions. The article includes 5 appendixes: a summary of the state of the art (A), descriptive data of the state of the art (B), summary of the experiment design (C), demographic data (D), and results of replications homogeneity (E).

2. Related Work

In this section, we review work related to MDD in empirical experiments through a Systematic Literature Review (SLR). Our **research question** when looking for related works was: What are the experiments in MDD, especially the experiments within a family? The **search string** used was: (family of experiments OR several experiments OR empirical evaluations OR empirical studies OR experiments OR replications) AND (model-driven OR MDD OR MDA OR MDE). The **inclusion criteria** were: (IC1) empirical experiments in MDD (even when no replications are reported); (IC2) articles that deal with conceptual models. The **exclusion criteria** were: (EC1) articles with no experiments; (EC2) articles without subjects; (EC3) articles that are not related with models; (EC4) articles that do not describe the results of the experiments. The **Quality** criteria to prioritize the results were: (QC1) the journal is in JCR list; (QC2) the validation is based on a family of experiments, (QC3) there are statistical significant results. The search was run in February 2021 on Scopus, IEEE Xplore and ACM Digital Library.

Appendix A shows a summary of all related works. For each work, we describe the goal, number of subjects, number of replications, variables, results and limitations. Most of these elements are described in the work of Ferreira et al. [13] as the basic information needed in software-engineering experiments. Appendix B shows the results of the quality criteria and the aggregation of results according to analysed variables, rank of subjects, MDD models and MDD tools used in the experiment. For each result we also show the percentage of articles that deal with it.

The comparison of MDD with a traditional method is not new in Software Engineering (SE). For example, among others we can reference the work of Martínez et al [14], who have compared Model-driven, Model-Based and Code-Centric with respect to their adoption among software developers. The experiment was conducted with 26 students to measure Perceived Usefulness, Perceived Ease of Use, Compatibility of Each Method and Intention to Adopt. The results show that MDD is the most useful, although it is also considered the least compatible with previous developers' experiences. Developers feel comfortable with the use of models, and they are likely to use them. The main limitation of this work is the small sample size (26 subjects), with no replications, which limits the generality of the results. Brdjanin et al. [15] conducted an experiment to evaluate databases generated from BPMN models in combination with UML class diagrams versus a manual design. There are no replications of this experiment. The analysed variables are Effectiveness (measured through recall and precision), Efficiency (measured as time) and Usability. The results show that the automatically generated database model can be considered as starting point for a manual design. The combination of MDD and a manual design provides the best result. The main limitation is that the time to train subjects was so long that from an initial group of 95 subjects, only 31 finally participated. Wanderley et al. [16] conducted an experiment to compare building conceptual models from scratch versus using mind maps as a first step. 18 subjects participated in the experiment with no replications. The analysed variables were Effort and Usability; the results conclude that the time spent producing conceptual models with the help of mind maps

was always less than that needed to produce models without maps. Usability is not analysed in detail; the analysis focuses on how easy mind maps are and their understandability.

Other groups of works have dealt with a family of experiments to evaluate MDD methods, such as Gonzalez-Huerta et al. [17]. That work aims to validate QuaDAI, a method for the derivation, evaluation and improvement of software architectures in MDD. The family is composed of 4 replications and a total of 92 subjects . The analysed variables are Effectiveness, Efficiency, Perceived Ease of Use, Perceived Usefulness and Intention to Use, with regard to using QuaDAI versus Architecture Tradeoff Analysis Method (ATAM). The results showed that when applying QuaDAI, the subjects obtained architectures with better values, and that they found the method easier to use, more useful and more likely to be used. The results of each replication are aggregated through meta-analysis. Each replication has a low sample size. Ricca et al. [18] conducted a family of 5 replications with 100 subjects. The goal was to study the benefits of MDD for maintainability with respect to a traditional approach. The comparison was operationalized with a state-based tool (UniMod) for MDD, and Java for the traditional development. The metrics used in the family are Correctness, Time Spent and Efficiency. The results show that the use of UniMod reduces efficiency and time, but there are no significant differences regarding accuracy. The main limitation is that these results can only be generalized for MDD tools based on state machine models. Other work of Ricca et al. [19] aims to analyse how developers' experience and ability influence web application comprehension. They conducted a family of 4 replications with 75 subjects to study the variable comprehension. This is measured in terms of precision and recall. Results indicate that subjects having different ability achieved different performance and different levels of benefit by using stereotyped diagrams. Low experience subjects benefited more from stereotyped diagrams. The main limitation is that the experiments are only based on the stereotypes of UML Web Application Extension. The work of Cruz-Lemus et al. [20] is a family of experiments with a baseline and two replications conducted with 91 subjects. The goal of that work is an understandability prediction model to know the complexity of statechart diagrams in UML notation. Results show that complexity factors that influence the understandability of statechart diagrams are control flow features, entry/exit actions, and number of activities. The main limitation of this work is that it is based on a correlation analysis, there is not a controlled experiment with different treatments. Reggio et al. [21] present two replications of an experiment with the goal of evaluating the comprehensibility of business processes through a style and the effort to comprehend that style. The experiment was conducted with 88 students from two different universities. Results show that all the subjects achieve a significantly better comprehension with the style, the use of the style did not have impact on the effort, and more experienced participants benefited more from the use of the style. As limitation of this family of experiments we highlight that problems used in the experiment are very easy, which may hinder more significant results.

Other works focus on analysing characteristics of MDD methods (not a whole MDD development process). Fernández et al. [22] conducted a family of experiments to evaluate a web usability evaluation process (WUEP) that measures usability in an MDD model. The study focuses on comparing WUEP versus a heuristic evaluation. The family is composed of 3 replications and 64 subjects. The metrics are Effectiveness, Efficiency, Perceived Ease of Use and Satisfaction. Each replication is analysed individually, and the aggregation is done through meta-analysis. The results show that WUEP is more effective, efficient and produces more developers' satisfaction than a heuristic evaluation. Heuristic evaluation is perceived as easier than WUEP. The main limitation of this family is that, prior to this evaluation, most of the subjects did not have a good knowledge of usability. Mussbacher et al. [23] have conducted an experiment based on design thinking to analyse the characteristics of MDD that justify why it has not been universally adopted in industry. The design thinking was done with 15 MDD experts, where each one had to identify the future challenges in MDD to be adopted by industry. The results show that the main MDD challenges for next 30 years are: Cross-disciplinary model fusion, personal model experience, flexible model integration, and resemblance modelling. The main limitations of this experiment are that the sample size is not large and there is a lack of quantitative metrics.

Other works compare different approaches to working with MDD. Cachero et al. [24] compares two types of domain model notations, graphical versus textual. The experiment recruited 127 subjects and the analysed variables were Efficiency and Effectiveness. The results demonstrate that the improvement of a graphical notation compared to a textual one is statistically significant. The limitations of this experiment are that the size of the models used were relatively small and in the same domain. Planas and Cabot [25] have conducted an experiment with 45 subjects to compare the usability of two modelling tools that can be used in MDD: MagicDraw and Papyrus. The evaluation is based on the variable Usability in the Modelling Process, and the modelling obstacles the subjects encounter during the process. The results show that there are no differences between both tools. The main limitation is that the experiment was based on videos; there was no discussion or questionnaires for the subjects. Safdar et al [26] conducted an experiment to compare three modelling tools: IBM rational Software Architect (RSA), MagicDraw, and Papyrus. The experiment recruited 30 subjects to measure productivity in terms of modelling effort required to correctly complete a task, learnability, time, number of clicks and memory load. The results show that RSA is better in terms of time and number of clicks when modelling class diagrams and state machines. The main limitation of the experiment is that it focuses only on three UML diagrams: class, state and sequence.

Finally, there are other works that aim to evaluate experiments in the software engineering area. Freire et al. [27] presents an empirical study which evaluates a Domain-

Specific Language (DSL) proposed to formalize experiments in software engineering. The experiment was conducted using as sample size of 16 experiments in the software engineering area evaluating Completeness and Expressiveness. The results highlight several limitations of the DSL which affect the formalization and execution of experiments. The main limitation of this experiment is that the authors of the DSL, themselves, conducted the experiment.

Our conclusions from the related works highlight that there are not many articles that have compared MDD methods among them ([17], [24], [25], [26]), and only one of them did this with a family of experiments ([17]). Several authors such as Basili [28] highlight the importance of replications to build a sound body of knowledge. Regarding the comparison of MDD versus a traditional method, this appears only in two articles ([14], [15]) with no replications. Most articles focus on measuring effectiveness [15] [17] [22] [24], efficiency and effort ([15], [16], [17], [18], [21], [22], [24]), since these benefits are usually claimed for MDD. Regarding the number of subjects, 7 articles ([17], [18], [19], [20], [21], [22], [24]) recruited more than 50 participants. The current article aims to cover the lack of families of experiments to compare a traditional method versus MDD, independently of a specific MDD tool.

3. MDD VS A TRADITIONAL METHOD

This section describes both development methods analysed in the family of experiments: MDD and a traditional method. Note that in both development methods, the developers have to develop fully functional systems from scratch. Since MDD can be operationalized into several tools, we opted for two different tools (INTEGRANOVA and WebRatio) in our family for the sake of generalizing the results as much as possible. This choice is due to the fact that both tools implement the MDD paradigm from a conceptual model programming point of view. In this case, the model is the code, so analyst must focus all their efforts on building conceptual models. Code is relegated to automatic code generation transformations that are hidden for developers. No code is manually written. Even though there are other tools that operationalize this approach of MDD, such as NDT [29], these other tools are mainly based on textual notation. Since both INTEGRANOVA and WebRatio are based strictly on graphical models and they are widely referenced in research articles of conceptual model programming [30], we opted for this choice.

3.1. INTEGRANOVA

INTEGRANOVA [10] is an MDD tool that generates code in Java and C# (both for Web and Desktop) from conceptual models. Each conceptual model represents a different characteristic of the system:

- The Object Model specifies the system structure in terms of classes of objects and their relationships. It is modelled as an extended UML class diagram.
- The Dynamic Model represents the sequences of events that can occur for a class of objects, and the interaction between object classes.
- The Functional Model specifies how events change object states. The behaviour of the system is modelled by the Functional and Dynamic Models working together.
- The Interaction Model represents the interaction between the system and the user.

3.2. WebRatio

WebRatio [11] is another MDD tool based on Eclipse that generates web and mobile applications from conceptual models. Below, we describe these models:

- The Domain Model is a conceptual schema that represents all the classes with their properties and relationships needed in the system. It is represented with UML class diagram notation.
- The Interaction Flow Modelling Language (IFML) [31] represents the system interfaces. This was adopted by the OMG as the standard to represent interfaces abstractly.
- The Action Model that specifies the CRUD (Create, Read, Update, Delete) operations that can be done in the system.

Note that the only point in common between INTEGRANOVA and WebRatio is the Domain Model. The syntax and the semantic of this model are the same in both tools. This is because both tools adopt the standard UML notation. Even though the goal of the other models is shared between both tools, they are completely different. There are no similar syntaxes to represent the functionality or the interfaces. Apart from the models, the process to generate code from the conceptual models is also different. In the case of INTEGRANOVA, there is a Web service that sends the models to a server that downloads the code. Next, the developer has to setup said code to run the system. The complete process takes around 10 minutes. For WebRatio, the process is shorter (around 1 minute). The model is compiled in the local machine and the code is automatically deployed when downloaded. Both INTEGRANOVA and WebRatio generate fully functional system, which is the target of our study. The developers' effort is focused on conceptual models, and the whole system is automatically generated.

3.3. Traditional Method

In a traditional method, the developer focuses on writing the code that implements the system. The implementation can be done in any programming language. In this method, the code is the only representation of the system. There are two main approaches:

- Code-centric: developers do not use any conceptual model; they just write the code directly from the list of requirements.
- Model-based: developers use at least one model (in any notation) to represent the parts of the system that are more confusing. The transformation from these models to code is done manually.

4. REPLICATIONS THAT COMPOSE THE FAMILY

All replications follow a within-subjects design where subjects have to develop a system from scratch, bot using a

traditional method and MDD (see summary in Appendix C). Differences among replications are the MDD tool (INTEGRANOVA and WebRatio) and the order of the training problems used to learn a traditional method and MDD. The aggregation of the replications for the analysis is done through two contextual variables: one to represent the replication and other to represent the MDD tool used. Next, we describe the details of the experiment design.

4.1. Common Characteristics among Replications

The hypotheses, factor, response variables, metrics, profile of subjects, design, experimental problems, experiment procedure and experimenters are the same for all replications. All these characteristics can be seen in [7], next we summarize them to provide a self-contained article.

The goal is to compare MDD versus a traditional software development method independently of the MDD tools, for the purpose of identifying differences and similarities between methods. We focus on system developed attributes as well as developers' workload. The experiment is conducted from the perspective of researchers and practitioners interested in MDD.

From all the benefits that the literature attributes to MDD (summary in [7]), we focus our family on the most referenced ones: Functional Suitability, Effort and Satisfaction. Next, we describe the experimental design according to Juristo and Moreno [32]. **The research questions** and **hypotheses** are:

- **RQ1:** Is Functional Suitability affected by MDD? According to ISO 25000 [33], Functional Suitability is "*The degree to which the software product provides functions that meet stated and implied needs when the software is used under specified conditions*". The null hypothesis tested to address this research question is: H_{01}: Functional Suitability of a system built using MDD is similar to Functional Suitability using a traditional method.
- **RQ2:** Is developer's Effort affected by MDD? According to IEEE [12], Effort is defined as "*the number of labour units required to complete a scheduled activity or work breakdown structure component, usually expressed as person-hours*". The null hypothesis tested to address this research question is: H_{02}: Developer's Effort to build a system using MDD is similar to Effort required using a traditional method.
- **RQ3:** Is developer's Satisfaction developing a system affected by MDD? Satisfaction is defined as the contentedness with and positive attitudes towards product use [12]. The null hypotheses to address this research question is: H_{03}: Developer's Satisfaction using MDD to build a system is similar to the level of Satisfaction using a traditional method.

The family of experiments analyses the **factor** development method, where the control is a traditional method and MDD is the treatment. As **response variables** we have Functional Suitability to answer RQ1, Effort for RQ2, and Satisfaction for RQ3. Functional Suitability is measured through the metric Effectiveness. According to ISO 25000 [33], Effectiveness is defined as "*The degree to which specified users can achieve specified goals with accuracy and completeness in a specified context of use*". We measure Effectiveness as the percentage of test cases successfully passed in the system developed with the treatment from scratch. Each test case is divided into several items (steps in the execution) in such a way that the addition of all the items that compose a test case is 1. The test case returns a value between 0 (no item has passed the test) and 1 (all items have passed the test), including decimals. The aggregation of all the test cases used in a system is calculated through the average. For example, if we have a test case with 10 items and only 8 items are passed successfully, the effectiveness of this test case is 0.8. If we have three test cases, with an effectiveness of 0.8, 0.6 and 0.5, the aggregation to calculate the effectiveness of the whole system is 0,63 ((0.8+0.6+0.5)/3). Effectiveness focuses on the functionality; usability of user interfaces is out of target.

Previous studies claim that MDD yields better effectiveness in complex problems compared to simple ones. So, we have divided Effectiveness into two metrics: **Effectiveness in Simple Problems** and **Effectiveness in Complex Problems**. In order to differentiate between simple and complex problems, we have divided each problem into three parts. The effectiveness obtained in the first part corresponds to simple problem, while the whole problem corresponds to complex problem. Subjects can only continue with the next part if the first one is completed. So, the same problem is used to measure both types of effectiveness.

Regarding Effort, this is measured as the **time** needed to complete the first part of the experimental problem (the part corresponding to simple problem). The experimental problems are too long to be completed in 4 hours (the maximum time to complete the experiment). So, we ensure that most subjects managed to complete this first part. If we analysed the time to develop the whole problem (complex problem), all subjects would have the same time, since all of them would need 4 hours, and we cannot analyse differences among treatments. Satisfaction is measured through three metrics: **Perceived Ease Of Use (PEOU), Perceived Usefulness (PU), and Intention To Use (ITU),** as proposed by Davis [34] . These three metrics were measured using a 5-point Likert scale questionnaire named MAM. Based on Moody [35], in the MAM questionnaire, we defined eight questions to measure Perceived Usefulness, six questions to measure Perceived Ease of Use and two questions to measure Intention to Use. We defined a questionnaire for each treatment (MDD and traditional method).

Even though the meaning of each question was the same for both levels, each questionnaire includes terms specific to the level that we aim to measure. For example, the statement "*I will definitely use MDD to develop web applications*" is used to measure Satisfaction with MDD, whereas the statement "*I will definitely use a traditional method to develop web applications*" is used to measure Satisfaction with the traditional method.

Note that when we refer to Functional Suitability, Effort and Satisfaction in this article, we refer to the meaning that the metrics previously described provide. All these response variables have more subtle aspects such as understandability, originality, elegance, etc. that are out of scope of our experiment.

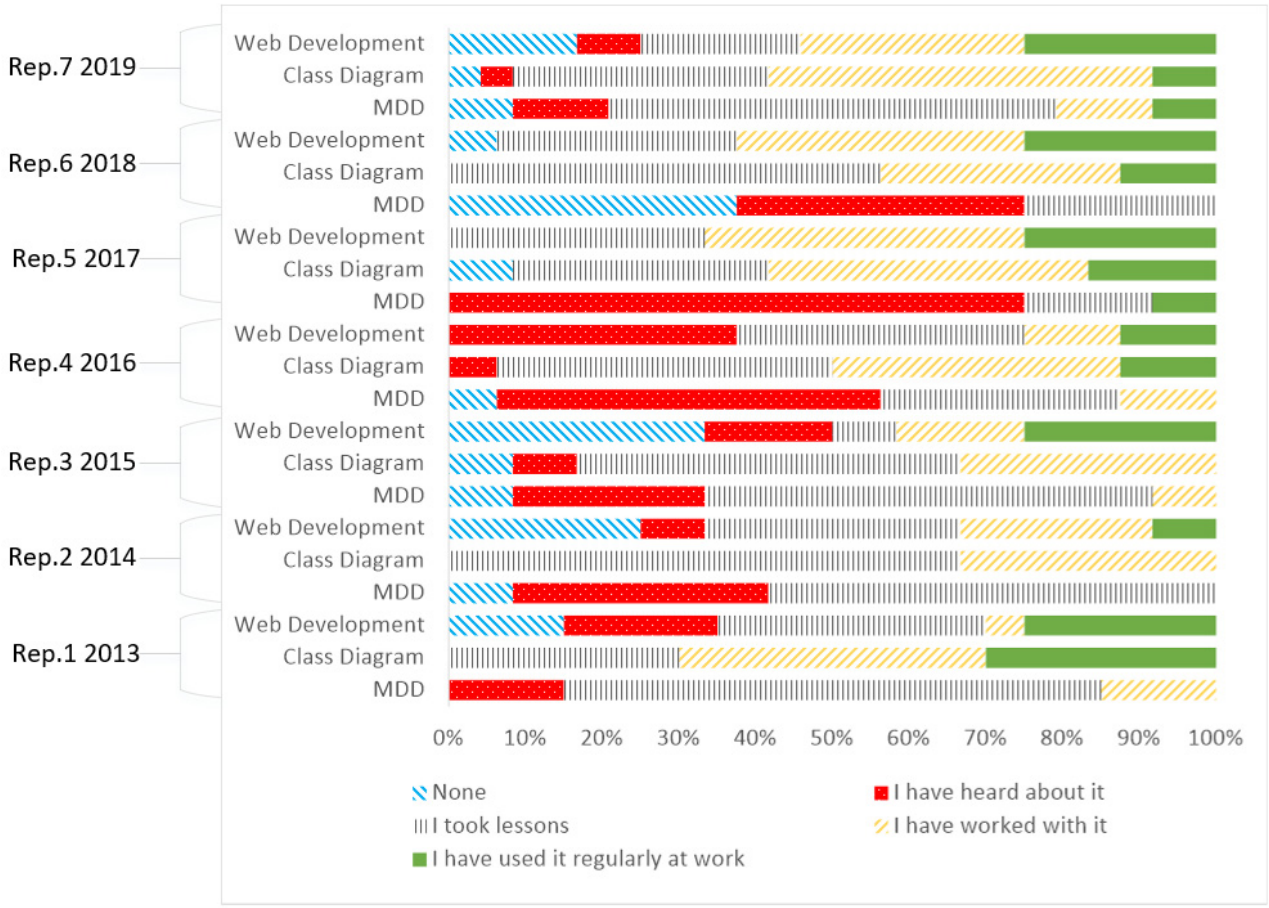

Figure 1. Previous experience with the applied treatments

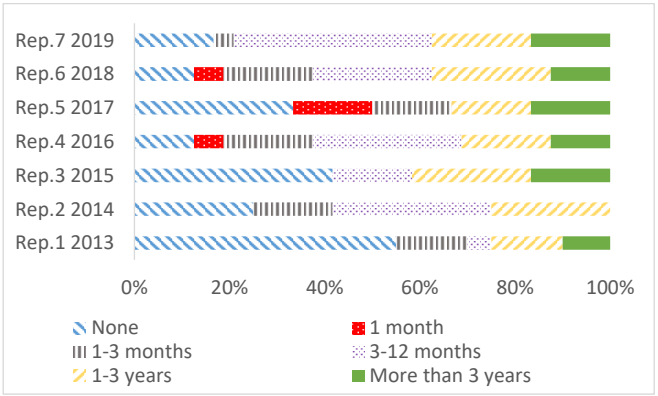

Figure 2. Job experience at software companies per replication

The **subjects** are Master´s degree students from the Universitat Politècnica de València (UPV, Spain) who are taking a course in software engineering, specifically in MDD. Even though they are students, most of them have real experience in software development companies, as Figure 2 shows. All of them had good knowledge of a traditional development, but only a few had heard something about MDD before this course (as Figure 1 shows). Raw data can be seen in Appendix D. The subjects participated in pairs in the experiment, so in our experiment a sample unit is a pair of subjects. The **experimental design** is a paired design blocked by experimental problems [32]. The sample units are divided into two groups (G1 and G2). Both groups started with the traditional development and moved on to MDD. The only differences between the groups were the experimental problems. G1 started with Problem 1 and went on to Problem 2 in MDD, while G2 did the opposite. This way we avoid a learning effect of the problem between treatments and we mitigate the threat of results that depend on a specific problem (both treatments work with both problems). We used a within-subjects design since it provides the largest possible sample size. All sample units participated in the experiment at the same time, so we avoided variability among different possible contexts.

The **experimental problems** are two necessarily simple problems that describe a system in such a way that the subjects had to develop it from scratch. One problem is the Invoice Problem, which aims to manage an electrical appliance company and the other problem is the Photographers Problem, which aims to manage a company that works with freelance photographers. The first part of the Invoice Problem (simple problem) has 100 function points, while the whole problem (complex problem) has 272 function points. For the Photographers Problem, we have 94 function points for the simple problem version and 199 for the complex problem version. These functions points have

been calculated from an experimenters' solution to give an idea of how complex problems are. Other subjects' solutions may differ slightly but not in problem magnitude.

The **experimental procedure** is made up of the following steps:

1. Subjects fill in a demographic questionnaire to report their previous knowledge of both treatments.
2. Sample units train with the traditional development (16 hours).
3. Sample units apply the treatment of the traditional method (4 hours).
4. Sample units train with the MDD method (16 hours).
5. Sample units apply the treatment of MDD (4 hours).
6. Sample units fill in open questions with 3 pros and cons of MDD to try to justify the results of the experiment.

The details of the problems, the test case items , and the steps that define the experimental procedure can be seen in [8]. The source materials used in the experiment can be seen in [36] for future replications.

4.2. Different Characteristics among Replications

The few differences that we have incorporated into the replications have been to generalize the results as much as possible. Table 1 summarizes these changes. For each replication, the table shows the academic year when the replication was conducted, the MDD tool used, the sample size (subject pairs in our experiment) and the training problems used in each method (V= Video club and T=Transport).

Table 1. Summary of changes among replications

Replication	MDD tool	Sample Size	Training Problems
Rep.1 2013	INTEGRANOVA	10	V/T
Rep.2 2014	INTEGRANOVA	6	T/V
Rep.3 2015	INTEGRANOVA	6	V/T
Rep.4 2016	INTEGRANOVA	8	T/V
Rep.5 2017	WebRatio	6	V/T
Rep.6 2018	WebRatio	8	T/V
Rep.7 2019	WebRatio	12	V/T

The MDD treatment has been operationalized through two **MDD tools**: INTEGRANOVA and WebRatio. The learning process for each one was different due to the particularities of each tool, but the experimental problems and the time spent were the same. This change aims to extract conclusions independently of the tool. The **sample size** also fluctuated depending on the number of students taking the Master's degree. This size does not have much statistical power if we analyse the results per each replication independently, but the power is sufficient if we aggregate the replications. We used two **training problems**, one for each treatment. For the replications we swapped the training problem used in each method. In replications 1, 3, 5 and 7 we used the Video club problem (a system to manage film renting) to train the traditional method and the Transport problem (a system to manage routes of public buses) for MDD. For replications 2, 4, 6 and 8 we interchanged the training problems. This change was done to ensure that the learning process did not depend on the training problem used. These training problems were not analysed as part of the experiment, they were only used to ensure that subjects had enough knowledge of both treatments.

4.3. Aggregation of the family of experiments

Our aggregation of replications is based on the work of [37], which proposes using a multilevel hierarchy. The aggregation is done through **contextual variables**; variables that reflect the differences between the experiments and which need to be further investigated. Figure 3 shows the multilevel hierarchy of the family of experiments. In the first level, we have the subjects of each experiment; in the second level the repeated measurement of each replication (we have a within-subjects design); in the third level we have the replication; and finally, in the fourth level we have the tool where the MDD method is operationalized in each replication. The study of differences in Level 1 (named **participant variables** in [37]) is not of interest in our study since we are not interested in analysing differences among subjects of each replication individually, but on the aggregation of replications.

So, we include subjects as a random variable in our statistical model, but it is not the target of our study. The variable in Level 2 is called **Design variable** in [37], i.e., a variable that is the target of the design, and it is defined in the design of each replication. This is the target of each replication, and it is termed Method. This variable has two treatments: a traditional development and MDD. The variables of Level 3 and Level 4 are called **Contextual variables** in [37] , i.e., variables which are defined for the aggregation of replications. In Level 3 we have the variable Replication, with one level per replication, and in Level 4 we have the variable MDD tool with two levels: INTEGRANOVA and WebRatio. Note that MDD tools are only used for the MDD treatment, so with the aggregation of replications we are interested in analysing possible differences between tools only for the MDD method. So we will analyse in detail Method*MDD tool interaction to study whether one Method yields better values depending on the MDD tool. Even though these contextual variables are not part of our experimental design, they are also a target of study since they allow us to identify differences between replications. In order to ensure that samples of each replication are homogeneous for the aggregation, we conducted a Chi-Square test. Appendix E shows the p-values of analysing the independence of the variable Replication with each response variable. Values higher to 0.05 means that samples are homogeneous. All p-values are higher to 0.05 except for Effort. This heterogeneity for Effort could be due to the contextual variable MDD tool, that may affect in the time of each replication. If we apply the Chi-Square test for the replications that worked with INTEGRANOVA, and we repeat the same test for replications that worked with WebRatio, we see that subjects are homogeneous in both groups of MDD tools. So, we can state that replications are homogeneous and there are not differences among replications that may have an impact on the results.

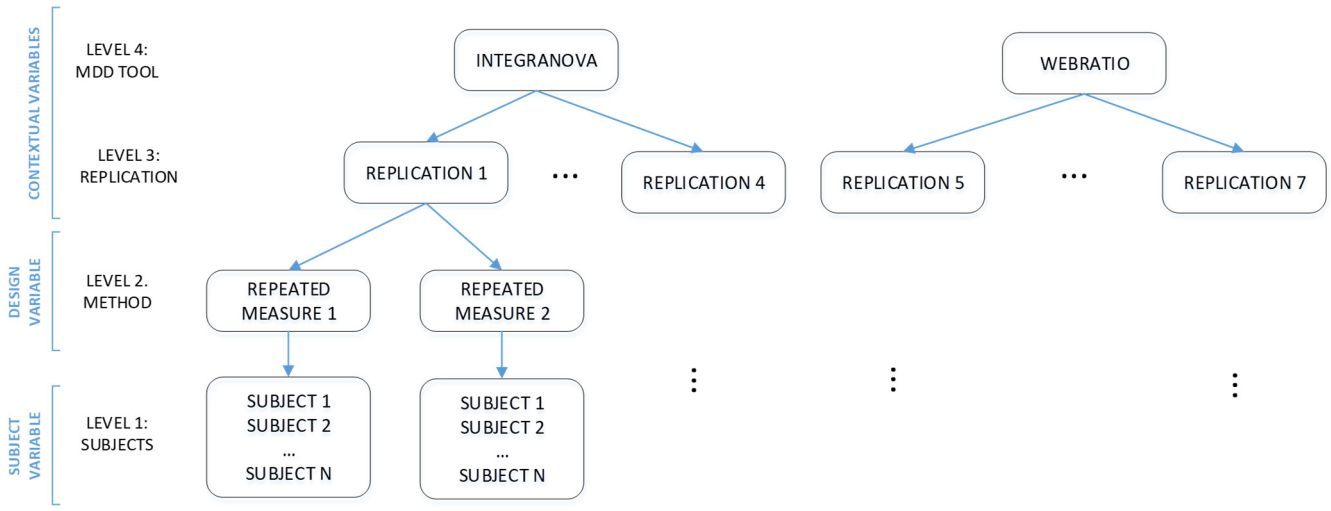

Figure 3. Multilevel hierarchy of the family of experiments

4.4. Data Analysis

We report descriptive data using box-and-whisker plots to illustrate the differences regarding the treatments of the design variable and the levels of the contextual variables. Descriptive data helps graphically identify possible differences between treatments or among levels.

As a statistical test to look for significant differences between treatments and among replications we use a mixed model, as proposed in [37]. The assumption for applying the mixed model is normality of residuals. The normality of residuals can be tested with the Saphiro-Wilk test applied to the residuals automatically calculated during the application of the mixed model test [38]. When the p-value is less than 0.05, we can reject the null hypothesis, which means that there are significant differences for the variable (design variable or contextual variable). In our analysis, we have defined Subjects (Level 1 in Figure 3) as a random variable, while Method (Level 2), Replication (Level 3), and MDD tool (Level 4) are defined as fixed variables. We used **scaled identity** as a covariance type for repeated measurements since it was the one with the lowest AIC (Akaike's Information Criterion) as proposed in [37]. We have also included the interactions of the design variable Method with the contextual variables. The analysis of these interactions is useful to study whether one treatment yields better values depending on the level of the contextual variables. Moreover, we have also analysed the blocking variable Problem and its interaction with Method to study whether there are also differences in treatments depending on the problem.

We have used Cohen's d [39] to calculate effect size in those variables with significant differences (variables whose p-value with the mixed model is less than 0.05). Cohen's d is defined as the difference between two means divided by a standard deviation of the data. According to Cohen [39], the meaning of the effect size is as follows: more than 0.8 is a large effect; from 0.79 to 0.5 is a moderate effect; from 0.49 to 0.2 is a small effect.

Using the mixed model, we cannot calculate power statistically (independently of the statistical tool used in the analysis). However, we used G*Power [40], finding that, for a repeated measurement statistical test, we need a sample size of 16 units for an effect size of 0.8 (large effect) to get a power of 80%. The sample size of each replication is less than 16 units, which implies a low power. So, we opt for analysing the aggregation of all replications, not each replication independently.

5. Analysis of Results

This section analyses the results according to the data analysis previously described for each response variable (descriptive data, mixed model, and Cohen's d). Raw data can be seen in [41]. Next, we study the descriptive data of each variable of our hierarchical multilevel tree (Figure 3), from the lowest level to the highest one.

5.1. Functional Suitability

Functional Suitability is measured through two metrics. Effectiveness in Simple Problems and Effectiveness in Complex Problems. Note that Effectiveness in Simple Problems is the effectiveness of one third of the problems, while Effectiveness in Complex Problems is the effectiveness of the whole problem. Since the experiment had 4 hours to be finished, just a few subjects managed to finish the whole problem; however, most of them finished the first third. Figure 4 shows the box plot for the metric **Effectiveness in Simple Problems** considering the design variable Method. Note that the line that connects both treatments in the plot shows the averages of each treatment. We appreciate that even though the median and the third quartile are similar for both treatments, the first quartile is better for MDD[1]. So, for simple problems, MDD seems to yield better values for effectiveness.

Figure 5 shows the box plot of Effectiveness for Simple Problems for the contextual variable Replication. We see a

[1] Number in the outlier means the sample id that differs significantly from other observations

high fluctuation of medians depending on the replication. This high variability could be because in each replication the sample size is small (around 6 sample units). This justifies our decision to avoid the analysis of each replication individually, and focus on the aggregation to extract conclusions.

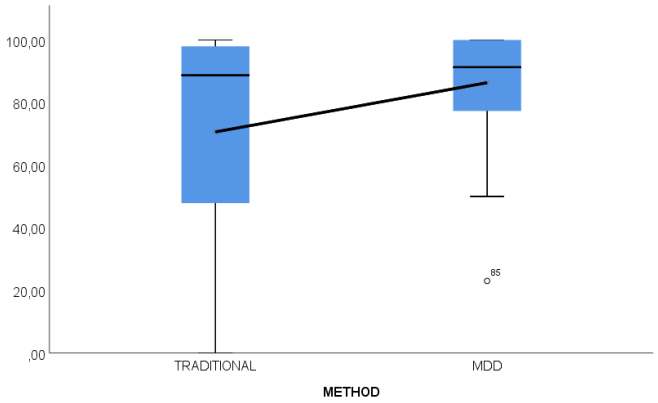

Figure 4. Effectiveness of Method considering simple problems

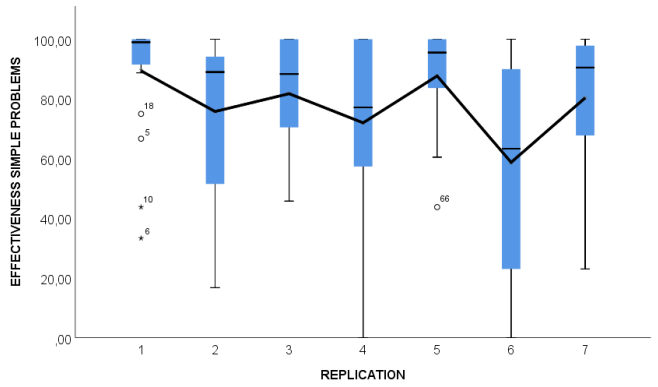

Figure 5. Effectiveness of Replication considering simple problems

Figure 6 shows the box plot of Effectiveness for Simple Problems for the MDD tool. First, median and third quartile are very similar, so we can state that there are no differences in effectiveness for simple problems for both MDD tools.

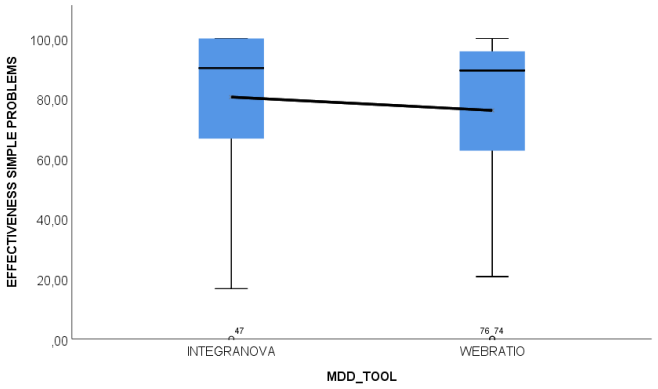

Figure 6. Effectiveness of MDD tool considering simple problems

Table 2 shows the p-values and the effect sizes (when the p-value is less than 0.05) of the variables. The results show that MDD is significantly better, and we also identify significant differences for the blocking variable (Problem). Note that the effect size for Method is low, and for Problem we have a large effect. Analyzing descriptive data, we can state that the Invoice Problem yields better values for effectiveness. Even though both problems have similar difficulty and size, the first problem has some constraints in the requirements that could complicate the design. These constraints consist of conditions that must be satisfied to assign photographers to reports and the period of time required to present two requests of the same photographer. We do not identify any differences in level 3 and 4 contextual variables.

Table 2. p-values[2] and effect size of the variables considering effectiveness in simple problems

Design variable	**Method**	**p-value**	**0.000**
		Effect size	0.104
Contextual variable	Replication	p-value	0.080
	Method*Replication	p-value	0.617
	MDD tool	p-value	0.358
	Method*MDD tool	p-value	0.976
Blocking variable	**Problem**	**p-value**	**0.000**
		Effect size	0.829
	Method*Problem	p-value	0.108

Figure 7 shows the box plot for **Effectiveness in Complex Problems**. We find that first quartile, median, and third quartile are better for MDD. The differences between medians are more pronounced than in the case of simple problems (Figure 4).

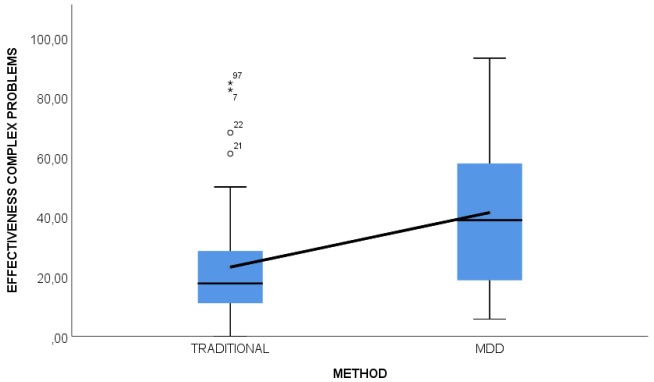

Figure 7. Effectiveness of Method considering complex problems

Figure 8 shows the box plot for Replication analyzing complex problems. Except for the first replication, all the medians follow the same pattern around 25%. If we compare these results with the results of simple problems, we notice that effectiveness decreases considerably (medians in simple problems were around 75%). This is because the subjects did not have enough time to complete complex problems in the 4 hours the experiment lasted.

[2] P-values are rounded to 3 decimals, that is why we have p-values=0

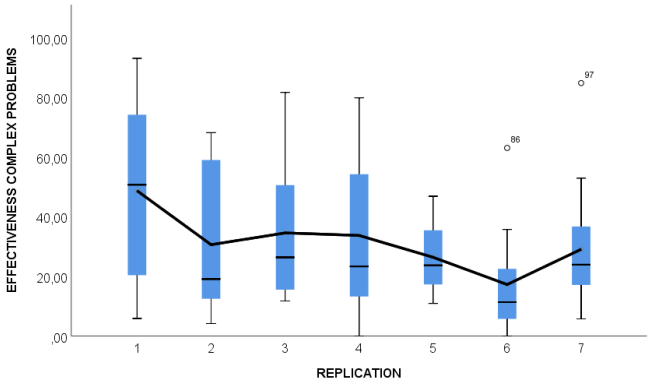

Figure 8. Effectiveness of Replication considering complex problems

Figure 9 shows the box plot for the MDD tool for complex problems. Median and third quartile are better for INTEGRANOVA, but these differences seem to be low.

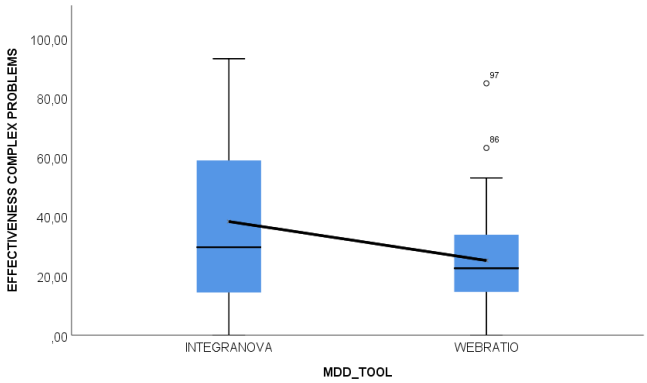

Figure 9. Effectiveness of MDD tool for complex problems

Table 3 shows p-values and effect sizes for effectiveness when the problems are complex. We appreciate significant differences for Method and MDD tool. The effect for Method is large, so MDD shows clearly better values for effectiveness in complex problem compared to a traditional method. The effect for the MDD tool is very low.

Table 3. p-values and effect size of the variables considering effectiveness in complex problems

Design variable	**Method**	**p-value**	**0.000**
		Effect size	0.984
Contextual variable	Replication	p-value	0.109
	Method*Replication	p-value	0.097
	MDD tool	**p-value**	**0.004**
		Effect size	0.321
	Method*MDD tool	**p-value**	**0.027**
Blocking variable	Problem	p-value	0.307
	Method*Problem	p-value	0.714

The results also show a significant difference for the interaction Method*MDD tool. Figure 10 shows the profile plot of this interaction. This plot helps us identify the tool that yields the best effectiveness for a specific method. When lines in the profile plot are not in parallel, we can state that there is a combination of Method-MDD tool that differs from the other combination. We note that INTEGRANOVA yields better effectiveness only in the MDD method, while in the traditional method both tools yield a similar effectiveness. This result makes sense, since MDD tools only affect the MDD treatment (not the traditional one), so differences appear only on the MDD treatment.

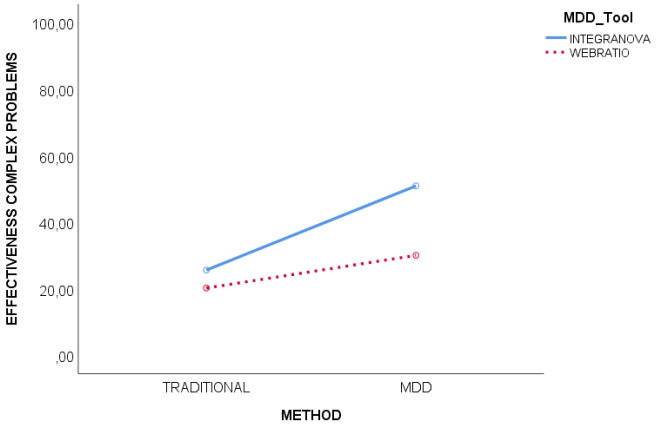

Figure 10. Profile plot of the Method*MDD tool for effectiveness in complex problems

So, to conclude, we can reject H_{01} (*Functional Suitability of a system built using MDD is similar to Functional Suitability using a traditional method*). MDD yields better Functional Suitability independently of the problem complexity and independently of the MDD tool.

5.2. Effort

Figure 11 shows the box plot for Effort for the variable Method. Note that Effort is measured as the time needed to complete the first exercise in the problem, since it is the only one that most subjects managed to finish within the time spent in the experiment. We see that MDD yields a better time than the traditional method. Both first quartile and median obtain better results with MDD.

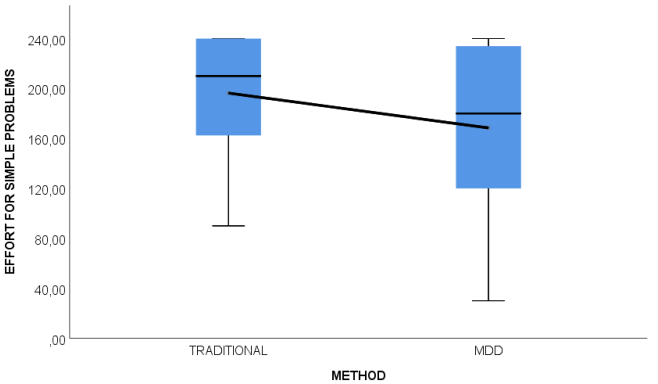

Figure 11. Time of Method

Figure 12 shows the box plot with the time in each replication. There is a high fluctuation among replications, perhaps due to the low sample size of each replication, individually.

Figure 13 shows the box plot with the time for each MDD tool. Medians yield more time in the use of WebRatio compared to the use of INTEGRANOVA, but the averages are the same for both tools.

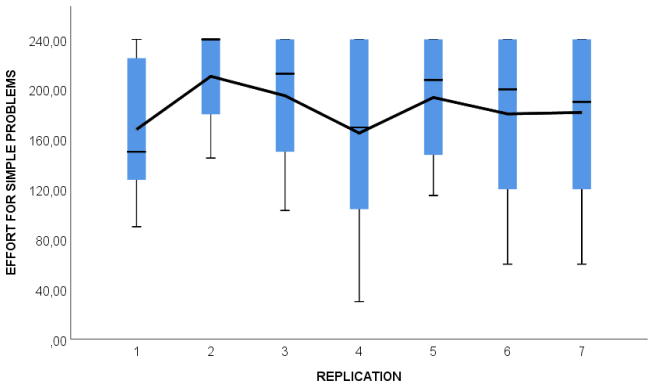

Figure 12. Time in each replication

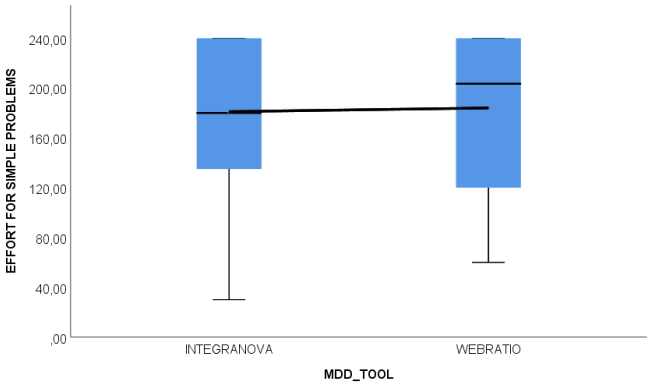

Figure 13. Time of MDD tool

Table 4 shows the p-values of the variables considering time. Method yields significant differences with a moderate effect. MDD requires less time than a traditional method. We also identify significant differences in two interactions: Method*MDD tool and Method*Problem.

Table 4. p-values and effect size of the variables considering time

Design variable	**Method**	**p-value**	**0.000**
		Effect size	0.532
Contextual variable	Replication	p-value	0.244
	Method*Replication	p-value	0.346
	MDD tool	p-value	0.831
	Method*MDD tool	**p-value**	**0.034**
Blocking variable	Problem	p-value	0.059
	Method*Problem	**p-value**	**0.005**

Figure 14 shows the profile plot for Method*MDD tool. This plot aims to show which tool yields the best Effort for a specific method. We see in the plot that there is a greater difference in time with the MDD tool compared to the traditional method. This is because the traditional method is the same for both tools, the differences may only arise in the MDD treatment. Figure 15 shows the profile plot of Method*Problem. The Invoice Problem is more sensitive to the treatment; MDD reduces its time almost to half. This could be because the Photographers Problem included several constraints which may make comprehension of the problem slightly more difficult.

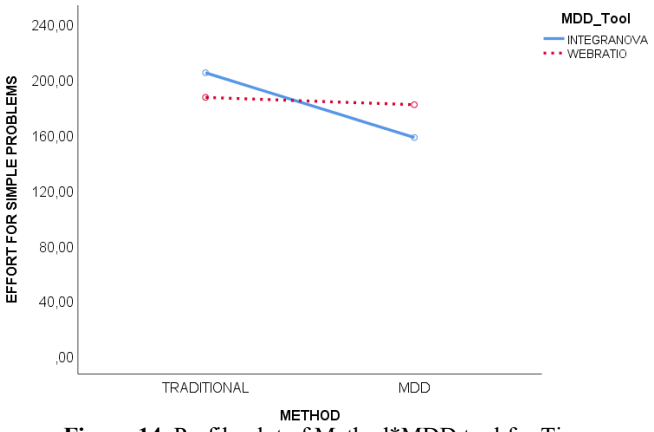

Figure 14. Profile plot of Method*MDD tool for Time

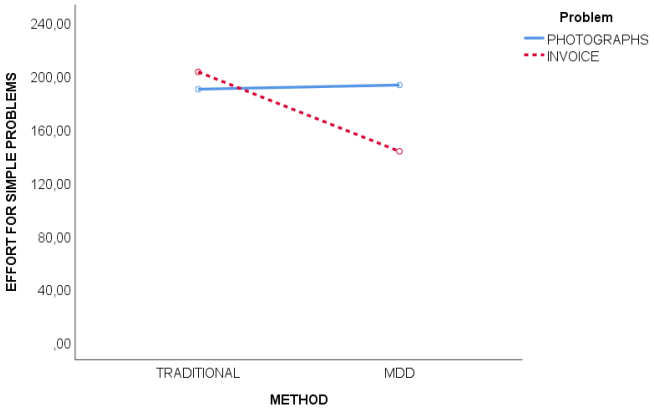

Figure 15. Profile plot of Method*Problem for Time

So, to conclude, we can reject H_{02} (*Developer's Effort to build a system using MDD is similar to Effort required using a traditional method.*) Effort for simple problem with MDD is less than when working with a traditional method, and INTEGRANOVA needs less Effort than WebRatio.

5.3. Satisfaction

Satisfaction is measured through three metrics: Perceived Ease of Use, Perceived Usefulness and Intention to Use. For the sake of brevity, we focus the descriptive analysis on the variable Method, which is the factor in our experimental design. The analyses of Problems, Replications, and MDD tools do not provide relevant results for our study.

Figure 16, Figure 17, and Figure 18 show the box plots for Perceived Ease of Use, Perceived Usefulness and Intention to Use regarding Method respectively. Only Intention to Use yields differences between both treatments, where the traditional method yields better first quartile, median, and third quartile. Table 5 shows the p-values for the variables considering Perceived Ease of Use. The results only yield significant differences for Replication, with a small effect. Analyzing descriptive data of replications, we see that replications with WebRatio yield slightly better results for Perceived Ease of Use than replications with INTERGRANOVA. However, these differences are so low that the variable MDD tool does not show significant differences.

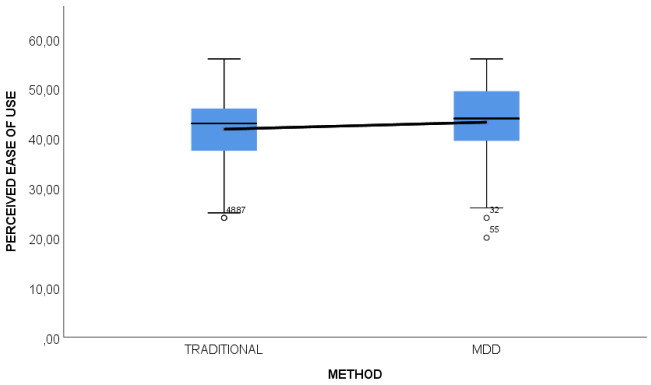

Figure 16. Perceived Ease of Use for Method

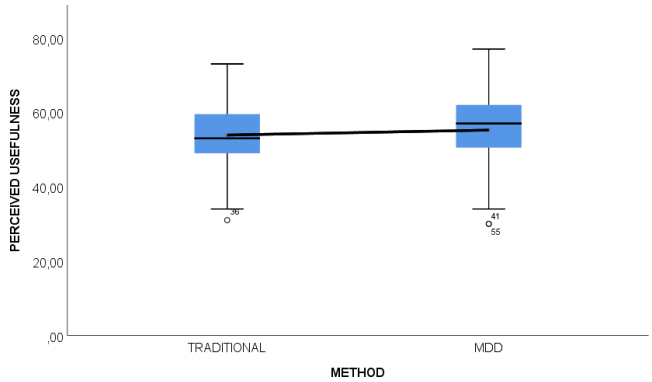

Figure 17. Perceived Usefulness for Method

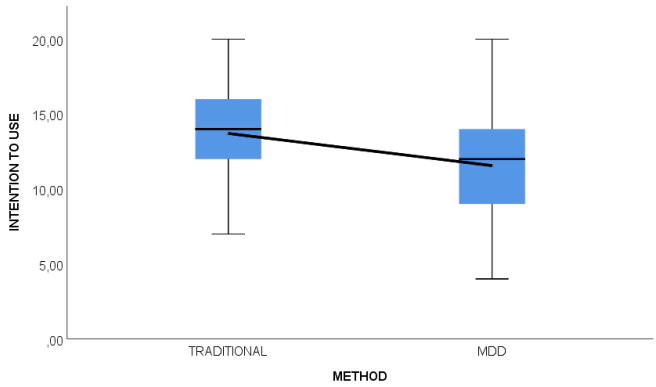

Figure 18. Intention to Use for Method

Table 5. p-values and effect size of the variables considering Perceived Ease of Use

Design variable	Method	p-value	0.346
Contextual variable	**Replication**	**p-value**	**0.044**
		Effect size	0.443
	Method*Replication	p-value	0.955
	MDD tool	p-value	0.125
	Method*MDD tool	p-value	0.317
Blocking variable	Problem	p-value	0.262
	Method*Problem	p-value	0.322

Table 6 shows p-values of the variables for Perceived Usefulness. The only significant difference appears in the interaction Method*Problem. Figure 19 shows the profile plot of this interaction. We appreciate that differences in problems are more important for the MDD treatment, where the Invoice Problem yields the best value. Again, the fact that this problem has no constraints may lead to perceiving MDD as easier (and therefore more useful) than in the case of developing the Photographers Problem.

Table 6. p-values and effect size of the variables considering Perceived Usefulness

Design variable	Method	p-value	0.541
Contextual variable	Replication	p-value	0.720
	Method*Replication	p-value	0.993
	MDD tool	p-value	0.142
	Method*MDD tool	p-value	0.129
Blocking variable	Problem	p-value	0.637
	Method*Problem	**p-value**	**0.043**

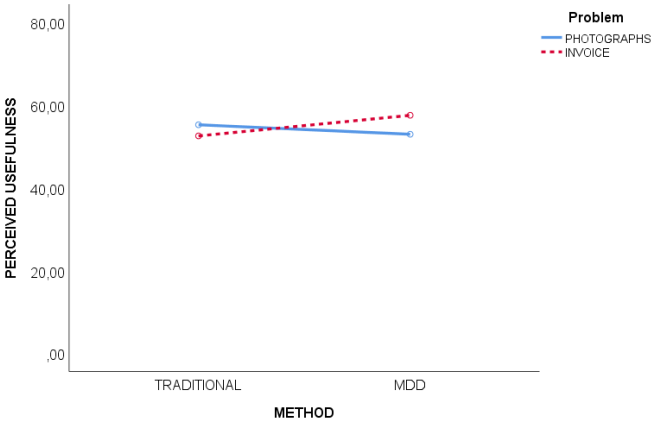

Figure 19. Profile plot of Method*Problem for Perceived Usefulness

InTable 7 we identify a significant difference for Method with a moderate effect. The traditional method yields a better Intention to Use compared to MDD. This leads us to think that even though MDD reports better values for effectiveness and time, subjects still prefer the traditional method. Figure 20 shows the profile plot for the interaction Method*MDD tool. The Intention to Use MDD is similar for both INTEGRANOVA and WebRatio. The differences appear in the traditional method, where the use of a specific MDD tool does not apply.

Table 7. p-values and effect size of the variables considering Intention to Use

Design variable	**Method**	**p-value**	**0.001**
		Effect size	0.604
Contextual variable	Replication	p-value	0.696
	Method*Replication	p-value	0.223
	MDD tool	p-value	0.097
	Method*MDD tool	**p-value**	**0.007**
Blocking variable	Problem	p-value	0.659
	Method*Problem	p-value	0.076

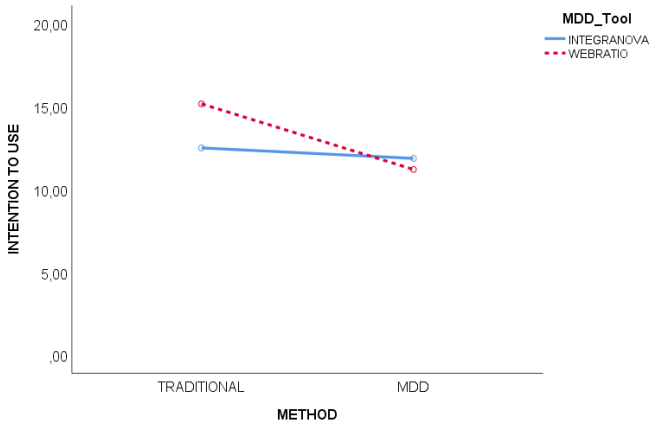

Figure 20. Profile plot of Method*MDD tool for Intention to Use

So, to conclude, we can reject H_{03} (*Developer's Satisfaction using MDD to build a system is similar to the level of Satisfaction using a traditional method*) only for the metric Intention to Use. The subjects intend to use a traditional method rather than MDD.

5.4. Threats to Validity

We have classified the threats that our aggregation of replications may be open to according to the classification provided by Wohlin [42]. We organized the threats of each type based on three groups: avoided, incurred and mitigated.

Conclusion validity. This threat is concerned with issues that affect the ability to draw the correct conclusions about relationships between the treatment and the outcome. Threats of this type are: (1) Low statistical power: this appears when the sample size is low and we are unable to reject an erroneous hypothesis. We avoided this threat by analysing the family of experiments as a whole through moderator variables. According to G*Power, we need a minimum of 16 sample units, and after the aggregation we have 56 units, which exceeds this limit. (2) Fishing: this appears when experimenters may influence the results by looking for a specific outcome. Our family experiences this threat since the experimenters themselves conducted the replications and the aggregation. (3) Reliability of measurements: this appears when measurements can show the wrong values. We have mitigated this threat, since Functional Suitability is measured by the experimenters, which reduces the errors, and Satisfaction is measured automatically via a Likert questionnaire. This threat may appear for Effort, since each sample unit must write down when she/he finished the first part of the problem to measure time. It is possible that a few sample units made mistakes when reporting time. (4) Reliability of treatment implementation: this appears when the implementation is not similar between treatments. We avoided this threat, since both treatments in all replications were based on the same instruments and procedure. (5) Random heterogeneity of subjects: this appears when sample units are highly heterogeneous. We avoided this threat, since, as the subjects' profiles show (Figure 1 and Figure 2) the sample units have a similar profile. (6) Previous subjects' experience, this threat appears when subjects do not have the same level of experience with both treatments. In our case, subjects are experts at a traditional method, but not experts at MDD. We mitigated this threat through lessons and training with MDD before the experiment. (7) Family wise error rate, which is the probability of coming to at least one false conclusion in hypothesis tests. This threat is avoided in our experiment since we are not conducting multiple tests (we do not have a general hypothesis that depends on more specific hypotheses). Thanks to the aggregation of the replications, we are analyzing just one sample test per metric (not a family of tests), so we have a wise error rate equals to the α of the null hypotheses (0.05).

Internal validity. This threat is concerned with influences that may affect the dependent variable with respect to a causality which the researchers are unaware of. Threats of this type that may appear are: (1) History: this appears when treatments are applied at different times. We mitigated this threat, since both treatments were applied under the same circumstances (the same instruments and the same experimenters). Even for replications spread across 7 years, no modifications were applied (same structure, same instructors, and same course content) (2) Maturation: this appears when subjects react differently over time. Our family of experiments experiences this threat since the MDD treatment is always applied after the traditional method. This choice is because, from a pedagogical point of view, we were only able to work with MDD after training with it. Even though this threat is impossible to avoid in an experiment embedded in a course where subjects have to learn (they must mature), we mitigated it using different problems in both treatments. (3) Instrumentation: this appears when errors in the instruments may affect the results. We mitigated this threat since the instruments were used in 7 replications, and the experimenters did not notice any error. Even though our satisfaction questionnaire is not validated, it is based on the TAM of Davis [34], which has been validated previously. (4) Mortality: this appears when subjects abandon the experiment. We avoided this threat since in all 7 replications none of the subjects abandoned the experiment. (5) Selection: this appears when subjects have not been chosen at random but for convenience. We have mitigated the dependency between subjects and results aggregating the several replications. (6) Ceiling effect: this occur when a high proportion of subjects in a study have maximum scores on the observed variable. We solved this threat with problems too complex to be solved in the time spent in the experiment. So, subjects reached the maximum level of effectiveness rarely. In order to solve this threat for Effort, we measured the time to finish simple problems instead of measuring the whole time (that is limited). (7) Suitability of subjects for experimental tasks: this appears when subjects play a role in the experiment in which they are not experts. Our experiment incurs this threat since we are asking engineers to do the work of a consultant.

Construct validity. This threat is concerned with generalizing the results of the experiment to the concept, or theory, behind the experiment. Threats of this type that our family may be open to are: (1) Inadequate pre-operational

explanation of constructs: this appears when the theory being analysed is not clear. We suffer this threat since metrics for Functional Suitability, Effort and Satisfaction reduce the analysis to passed test, time and Perceived Ease of Use-Perceived Usefulness-Intention to Use respectively. This may simplify the real world omitting other aspects such as usability or customization. (2) Mono-operation bias: this appears when the experiment deals with only a single factor, which may not give a full picture of the theory. We have mitigated this issue thanks to the contextual variables Replication and MDD Tool. (3) Confounding constructs and levels of constructs: this appears when there are different levels of expertise among subjects. We incurred this threat since, as Figure 1 shows, we have different levels of knowledge concerning MDD and the traditional method. Note that even though there are different levels, most subjects have a high knowledge of a traditional method and a low knowledge of MDD. (4) Experimenter expectancies: this appears when experimenters can bias the results based on what they expect. We have mitigated this threat by recruiting subjects with no special interest in the results of the experiment. (5) Problem homogeneity: this appears when experimental problems are too homogeneous to generalize the results to other problems. Our experiment suffers this threat since even though both problems refer to different contexts, both of them are from the same domain (information management domain).

External validity. This threat is concerned with conditions that limit our ability to generalize the results of our experiments to industrial practice. Threats of this type are: (1) Interaction of selection and treatment: this appears when subjects are not representative of the population we want to generalize. We mitigated this threat by recruiting subjects in the field of software development. Note that this threat is not completely avoided since the field of software development is highly diverse. Even though we have subjects with and without experience in the real world, only 14 subjects from 112 had three or more years of job experience. So we cannot state that our results are generalizable for high experience analysts. (2) Interaction of setting and treatment: this appears when the experimental material, or setting, is not representative of the target of study. We suffer this threat in two ways: (i) problems are necessary simple to be as much finished as possible to be runnable in 4 hours, so they are not representative of real complex problems. (ii) Only two MDD tools are evaluated, while the concept of MDD is wide. We have just focused our study on MDD tools based on a conceptual model programming perspective, where all analysts' effort is to build models (the full code is generated automatically with no human intervention). So generalization only applies to problems to be solved in 4 hours and for conceptual model programming MDD tools. (3) Interaction of history and treatment: this appears when the time when the experiment is conducted may affect the results. We have mitigated this threat using the same procedure, materials and experimenters in all replications. (4) Restricted generalizability: this appears when results can only be generalizable to a specific context. Our experiment suffers this threat since INTEGRANOVA and WebRatio define two methods that may be different from those used by other MDD tools. So our results cannot be generalized to MDD tools with methods very different from those used by INTEGRANOVA and WebRatio. Moreover, the context of Functional Suitability, Effort and Satisfaction is wide, and this experiment focus on passed test, time and MAM questionnaire respectively. So results can only be generalizable in those terms.

6. Discussion

This section discusses the results calculated previously for each variable, and the pros and cons extracted from the answers to the open questions concerning MDD written by the subjects at the end of the experiment. Even though replications are homogeneous in terms of the experimental design, we appreciate slight differences among replication samples. For example in Figure 5, Figure 8, and Figure 12 we see some differences in the medians. These differences could be due to the low power of each replication alone, which involves more variability in the replication. This is why we focus our analysis on the aggregation of all the replications, to have enough power to extract significant conclusions. The results for **Functional Suitability** are measured through two metrics: Effectiveness in Simple Problems and Effectiveness in Complex Problems. Effectiveness in Simple Problems shows that MDD is significantly better than a traditional method, although these differences are not much greater for complex problems. The values reported in both MDD tools are very similar, but we identified some differences between problems. The Invoice Problem yields better effectiveness than the Photographers Problem. The reason for this result could the lack of constraints in the Invoice Problem. This result reinforces the results of previous studies [9] that state that differences between a traditional method and MDD appears both in complex and simple problems, even though these differences are more evident in complex ones. Although there are differences in the way in which analysts work with INTEGRANOVA or WebRatio, these technical issues do not affect the results for effectiveness.

The results for Effectiveness in Complex Problems reinforce the existence of differences between a traditional method and MDD when the analyzed problems are complex, as in [9]. In this case, we also experienced differences between the MDD tools; WebRatio yields better values for effectiveness than INTEGRANOVA. To justify this difference, we analyzed the list of pros and cons concerning MDD which we asked the subjects to write about after the application of the MDD treatment. Most of the subjects who worked with INTEGRANOVA claim that the deployment of the model into a functional system requires too much time (around 10 minutes). Once the code is generated in C#, the database has to be generated through scripts, and the server and the client have to be compiled from the generated C# code. However, the deployment with WebRatio was much faster (around 1 minute). The system is automatically deployed on the cloud without scripts or compilations. This difference may justify that subjects using INTEGRANOVA had fewer opportunities to deploy their systems, and therefore carried out fewer tests,

so Functional Suitability could be affected by this issue. Not every change in the model could be quickly tested in a running system.

The results for **Effort** show that working with a traditional method requires more Effort for simple problems than working with MDD. If we combine this result with the Functional Suitability result, we can state that MDD yields more Functional Suitability with less Effort for simple problems than a traditional method. We see that INTEGRANOVA requires less Effort for simple problems than WebRatio, even though the deployment of the system in INTEGRANOVA requires more time. Looking at the open questions regarding the pros and cons of MDD, we see that many subjects who worked with INTERGRANOVA stated, as an advantage, that GUIs are automatically generated with no need for modelling. However, in the case of WebRatio, there is no default GUI model. The subjects had to model the GUI before running the system. So, this difference when dealing with GUI modelling in both MDD tools may justify the difference in Effort that appears for each tool. The problems also show differences; the Photographers Problem involves more Effort for simple problems than the Invoice Problem. Again, the existence of several constraints in the Photographers Problem may justify this difference.

The results for **Satisfaction** are divided into Perceived Ease of Use, Perceived Usefulness and Intention to Use. Only Intention to Use yields significant differences for the method, showing that subjects have the Intention to Use a traditional method rather than MDD. This result contradicts the results for Functional Suitability and Effort. Although MDD yields better Functional Suitability and Effort, the subjects still prefer the traditional development. In order to look for answers to justify this fact, we analyzed the open questions regarding the cons of MDD. The most repeated issue is that MDD does not seem as flexible as a traditional method. Other issues reported by the subjects are the following: MDD tools do not support non-SQL databases and microservices; the community that uses MDD tools is not large; there are few options to personalize GUIs, and there are no MDD tools to work collaboratively. Note that all problems are related with the tool that implements the MDD paradigm, not with the paradigm itself. This leads us to think that, maybe, more powerful and flexible MDD tools could change the subjects' opinion regarding their intention to use MDD in the future. Note that subjects are much more familiar with a traditional development. So, they may present inertia to change. Demographic data of Figure 1 shows that only 11% of subjects had worked with MDD previously, while 56% of subjects have been working with a traditional method more than 3 months in a real company (Figure 2). Maybe, the lack of job opportunities in the area of MDD [43] in the real world could affect the poor intention to use MDD.

Even though the results for Satisfaction in MDD are not positive, via the open questions the subjects mentioned some benefits after their experience. The most relevant being: it is easy to use for non-programmers; development is quick; it is user friendly; it is very visual; the learning curve is short, and, the homogeneity of the development. These benefits lead us to think that the subjects also see MDD as a positive paradigm which offers possibilities that a traditional method does not. The subjects do not express any important differences in Satisfaction depending on the MDD tool; both tools yield similar values for Satisfaction, even though the models which each tool works with are quite different.

Comparing our results with the works of the related literature, we see that our work agrees in the fact that MDD improves Functional Suitability and Effort, as other works such as [15], [16], [17], [18], [21], [22], and [24] state. However, our results do not yield a better Satisfaction for MDD, as other works such as [14], [20], [22] do. Maybe, this result is because we are comparing the Satisfaction of working with a traditional method (where subjects are very comfortable) versus satisfaction in MDD (a method completely new). On the contrary, experiments done in the literature measure Satisfaction only in an MDD context, so results are not biases by subjects' experience. Main novelties of our family of experiments regarding previous works can be summarized in: (1) comparison of a traditional method versus MDD in the development of a fully functional system from scratch; (2) generalization of results through the use of two different MDD tools; (3) conclusions are extracted from 7 replications, which reduces the possibility of contextual threats that may affect a particular replication.

7. CONCLUSIONS

This article presents a family of experiments consisting of 7 replications to compare a traditional method versus MDD (under the context of conceptual model programming), combining two MDD tools: INTEGRANOVA and WebRatio. The sample size of the whole family is 56. Aggregation of the replications has been done through moderator variables defined from a multilevel hierarchy. We have used the course of the replication and the MDD tool as moderator variables. The response variables analyzed are Functional Suitability (measured as effectiveness in simple problems and effectiveness in complex problems), Effort (measured as time spent in simple problems) and Satisfaction (measured as Perceived Ease of Use, Perceived Usefulness and Intention to Use). The information extracted through these response variables was complemented with open questions, where subjects had to write 3 pros and cons of MDD.

The results of the family of experiments conclude that Functional Suitability and Effort for simple problems yield significant differences between a traditional method and MDD. MDD yields the best values for Functional Suitability. This difference is larger for complex problems. Regarding Satisfaction, a traditional method has more Intention to Use than MDD. The lack of flexibility of MDD tools could justify this result. Note that MDD tools only allow to generate what can be represented through models, while manual coding allows more customized developments.

We can conclude that even though MDD provides more Functional Suitability and less Effort for simple problems, existing MDD tools still have to evolve to offer more features for analysts, such as collaborative work, support to

non-SQL databases and more personalization. One of the positive ideas extracted from the open questions is that the learning curve for MDD is not steep. This is an important advance for the wide adoption of MDD.

We have learnt some experimental design lessons. First, metrics should be applied by a unique experimenter to ensure homogeneity. Second, experiments that spend several sessions may be affected by a different context in each session. Third, it is important to check the instruments before starting the experiment. Fourth, the context of the replications must be as similar as possible to aggregate data.

As future work, we plan to replicate the experiment in different universities with different experimenters and more MDD tools. We also plan to prepare a list of recommendations to improve MDD extracted from the experience of the experiments.

Acknowledgements

This work was developed with the support of the Spanish Ministry of Science and Innovation project DataMe (TIN2016-80811-P), PGC2018-097265-B-I00, and was co-financed by ERDF. It also has the support of Generalitat Valenciana with GISPRO project (PROMETEO/2018/176) and GV/2021/072.

References

[1] S. J. Mellor, A. N. Clark, and T. Futagami, "Guest Editors' Introduction: Model-Driven Development," *IEEE Software,* vol. 20, pp. 14-18, 2003.

[2] D. W. Embley, S. Liddle, and Ó. Pastor, "Conceptual-Model Programming: A Manifesto," in *Handbook of Conceptual Modeling*, ed: Springer, pp. 3-16, 2011.

[3] S. W. Liddle, "Model-driven software development," in *Handbook of Conceptual Modeling*, ed: Springer, pp. 17-54, 2011.

[4] B. Hailpern, Tarr, P., "Model-Driven Development: the Good, the Bad, and the Ugly," *IBM Syst. J.,* vol. 45, pp. 451-461, 2006.

[5] B. Selic, "The Pragmatics of Model-Driven Development," *IEEE software,* vol. 20, pp. 19-25, 2003.

[6] T. Weigert and F. Weil, "Practical experiences in using model-driven engineering to develop trustworthy computing systems," in *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06)*, 2006.

[7] J. I. Panach, S. España, Ó. Dieste, Ó. Pastor, and N. Juristo, "In search of evidence for model-driven development claims: An experiment on quality, effort, productivity and satisfaction," *Information and Software Technology,* vol. 62, pp. 164-186, 2015.

[8] J. I. Panach, O. Dieste, B. Marín, S. España, S. Vegas, O. Pastor, and N. Juristo, "Evaluating Model-Driven Development Claims with Respect to Quality: A Family of Experiments," *IEEE Trans. Software Eng.,* vol. 47, pp. 130-145, 2021.

[9] J. I. Panach, O. Dieste, B. Marín, S. España, S. Vegas, O. Pastor, and N. Juristo, "Evaluating Model-Driven Development Claims with respect to Quality: A Family of Experiments," *IEEE Transactions on Software Engineering*, pp. 130-145, 2018.

[10] INTEGRANOVA, "INTEGRANOVA Technologies: http://www.integranova.com," ed.

[11] M. Brambilla and P. Fraternali, "Large-scale Model-Driven Engineering of web user interaction: The WebML and WebRatio experience," *Science of Computer Programming,* vol. 89, pp. 71-87, 2014.

[12] IEEE, *IEEE standard computer dictionary. A compilation of IEEE standard computer glossaries.* Institute of Electrical and Electronics Engineers. New York, EE.UU., 1991.

[13] W. Ferreira, M. T. Baldassarre, S. Soares, B. Cartaxo, and G. Visaggio, "A Comparative Study of Model-Driven Approaches For Scoping and Planning Experiments," Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, Karlskrona, Sweden, pp. 78-87, 2017.

[14] Y. Martínez, C. Cachero, and S. Meliá, "MDD vs. traditional software development: A practitioner's subjective perspective," *Information and Software Technology,* vol. 55, pp. 189-200, 2013.

[15] D. Brdjanin, G. Banjac, D. Banjac, and S. Maric, "An experiment in model-driven conceptual database design," *Software & Systems Modeling,* vol. 18, pp. 1859-1883, 2019.

[16] F. Wanderley, D. Silveira, J. Araujo, A. Moreira, and E. Guerra, "Experimental Evaluation of Conceptual Modelling through Mind Maps and Model Driven Engineering," Cham, pp. 200-214, 2014.

[17] J. Gonzalez-Huerta, E. Insfran, S. Abrahão, and G. Scanniello, "Validating a model-driven software architecture evaluation and improvement method: A family of experiments," *Information and Software Technology,* vol. 57, pp. 405-429, 2015.

[18] F. Ricca, M. Torchiano, M. Leotta, A. Tiso, G. Guerrini, and G. Reggio, "On the impact of state-based model-driven development on maintainability: a family of experiments using UniMod," *Empirical Software Engineering,* vol. 23, pp. 1743-1790, 2018.

[19] F. Ricca, M. D. Penta, M. Torchiano, P. Tonella, and M. Ceccato, "How Developers' Experience and Ability Influence Web Application Comprehension Tasks Supported by UML Stereotypes: A Series of Four Experiments," *IEEE Transactions on Software Engineering,* vol. 36, pp. 96-118, 2010.

[20] J. A. Cruz-Lemus, A. Maes, M. Genero, G. Poels, and M. Piattini, "The impact of structural complexity on the understandability of UML statechart diagrams," *Information Sciences,* vol. 180, pp. 2209-2220, 2010.

[21] G. Reggio, F. Ricca, G. Scanniello, F. D. Cerbo, and G. Dodero, "A precise style for business process modelling: results from two controlled experiments," Proceedings of the 14th international conference on Model driven engineering languages and systems, Wellington, New Zealand, pp. 138-152, 2011.

[22] A. Fernandez, S. Abrahao, and E. Insfran, "Empirical validation of a usability inspection method for model-

driven Web development," *J. Syst. Softw.,* vol. 86, pp. 161-186, 2013.

[23] G. Mussbacher, D. Amyot, R. Breu, J.-M. Bruel, B. H. C. Cheng, P. Collet, B. Combemale, R. B. France, R. Heldal, J. Hill, J. Kienzle, M. Schöttle, F. Steimann, D. Stikkolorum, and J. Whittle, "The Relevance of Model-Driven Engineering Thirty Years from Now," Cham, pp. 183-200, 2014.

[24] C. Cachero, S. Meliá, and J. Hermida, "Impact of model notations on the productivity of domain modelling: An empirical study," *Information and Software Technology,* vol. 108, pp. 78-87 2018.

[25] E. Planas and J. Cabot, "How are UML class diagrams built in practice? A usability study of two UML tools: Magicdraw and Papyrus," *Computer Standards & Interfaces,* vol. 67, p. 103363, 2020.

[26] S. A. Safdar, M. Z. Iqbal, and M. U. Khan, "Empirical Evaluation of UML Modeling Tools–A Controlled Experiment," Cham, pp. 33-44, 2015.

[27] M. Freire, U. Kulesza, E. Aranha, G. Nery, D. Costa, A. Jedlitschka, E. Campos, S. T. Acuña, and M. N. Gómez, "Assessing and Evolving a Domain Specific Language for Formalizing Software Engineering Experiments: An Empirical Study," *International Journal of Software Engineering and Knowledge Engineering,* vol. 24, pp. 1509-1531, 2014.

[28] V. R. Basili, "The Role of Controlled Experiments in Software Engineering Research," in *Empirical Software Engineering Issues. Critical Assessment and Future Directions: International Workshop, Dagstuhl Castle, Germany, June 26-30, 2006. Revised Papers*, V. R. Basili, *et al.*, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 33-37, 2007.

[29] M. Jose Escalona and G. Aragon, "NDT. A Model-Driven Approach for Web Requirements," *IEEE Transactions on Software Engineering,* vol. 34, pp. 377-390, 2008.

[30] G. Rossi, M. Urbieta, D. Distante, J. M. Rivero, and S. Firmenich, "25 Years of Model-Driven Web Engineering. What we achieved, What is missing," *CLEI Electronic Journal,* vol. 19, pp. 5-57, 2016.

[31] OMG. *Interaction Flow Modeling Language (IFML) : http://www.ifml.org/*.

[32] N. Juristo and A. Moreno, *Basics of Software Engineering Experimentation*: Springer, 2001.

[33] ISO/IEC, "ISO/IEC 25000 - Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE," 2010.

[34] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Q.,* vol. 13, pp. 319-340, 1989.

[35] D. L. Moody, "The method evaluation model: a theoretical model for validating information systems design methods," European Conference on Information Systems (ECIS 03), Naples, Italy 2003.

[36] J. I. Panach. *Experiment material*. Available: https://www.uv.es/joigpana/Experiment_materials.pdf

[37] P. Riofrío, S. Vegas, and N. Juristo, "A Method for Aggregating Families of Experiments in Software Engineerings. ," *Empirical Software Engineering Journal,* Pending to publish.

[38] L. S. Meyers, *Applied multivariate research : design and interpretation / Lawrence S. Meyers, Glenn Gamst, A.J. Guarino*. Thousand Oaks: SAGE Publications, 2006.

[39] L. Cohen, *Statistical power analysis for the behavioral sciences*, 2nd. Edition ed.: Lawrence Earlbaum Associates, 1988.

[40] E. Erdfelder, F. Faul, and A. Buchner, "GPOWER: A general power analysis program," *Behavior Research Methods, Instruments, & Computers,* vol. 28, pp. 1-11, 1996.

[41] J. I. Panach, Ó. Pastor, and N. Juristo. (2021). *Family of experiments MDD VS Traditional*. Available: Mendeley Data, doi: 10.17632/kz5pn2whyd.1

[42] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*: Springer, 2012.

[43] J. Hutchinson, M. Rouncefield, and J. Whittle, "Model-driven engineering practices in industry," Proceedings of the 33rd International Conference on Software Engineering (ICSE), Waikiki, Honolulu, HI, USA, 2011.

Jose Ignacio Panach has been an assistant professor at the Universitat de València since 2011 and an Assistant Researcher at the Centro de Investigacion en Metodos de Produccion de Software (ProS) at the Universidad Politécnica de Valencia since 2005. Jose Ignacio holds a PhD in Computer Science (UPV 2010). His research activities focus on MDD, usability, and interaction modelling.

Oscar Pastor is Professor and Director of the Centro de Investigación en Métodos de Producciónde Software –ProS of the Universitat Politécnica de València. He got his PhD in 1992. Formerly he was a researcher for HP Labs, Bristol, UK. His research activities focus on web engineering, requirements engineering, information systems and MDD.

Natalia Juristo received her PhD degree from the Universidad Politecnica de Madrid (UPM) in 1991. She is currently a full professor of software engineering at UPM. She received a Finland Distinguished Professor Program (FiDiPro) professorship starting in January 2013. Her main research interests include experimental software engineering, requirements, and testing