Assessing the Effect of Screen Mockups on the Comprehension of Functional Requirements

FILIPPO RICCA, DIBRIS, University of Genova
GIUSEPPE SCANNIELLO, DiMIE, University of Basilicata
MARCO TORCHIANO, DAUIN, Politecnico di Torino
GIANNA REGGIO and EGIDIO ASTESIANO, DIBRIS, University of Genova

Over the last few years, the software engineering community has proposed a number of modeling methods to represent functional requirements. Among them, use cases are recognized as an easy to use and intuitive way to capture and define such requirements. Screen mockups (also called user-interface sketches or user interface-mockups) have been proposed as a complement to use cases for improving the comprehension of functional requirements. In this article, we aim at quantifying the benefits achievable by augmenting use cases with screen mockups in the comprehension of functional requirements with respect to effectiveness, effort, and efficiency. For this purpose, we conducted a family of four controlled experiments, involving 139 participants having different profiles. The experiments involved comprehension tasks performed on the requirements documents of two desktop applications. Independently from the participants' profile, we found a statistically significant large effect of the presence of screen mockups on both comprehension effectiveness and comprehension task efficiency. No significant effect was observed on the effort to complete tasks. The main pragmatic lesson is that the screen mockups addition to use cases is able to almost double the efficiency of comprehension tasks.

Categories and Subject Descriptors: D.2.1 [**Software Engineering**]: Requirements/Specifications

General Terms: Documentation, Experimentation, Human Factors

Additional Key Words and Phrases: Screen mockups, use cases, analysis models, controlled experiment, family of experiments, replicated experiments

ACM Reference Format:
Filippo Ricca, Giuseppe Scanniello, Marco Torchiano, Gianna Reggio, and Egidio Astesiano. 2014. Assessing the effect of screen mockups on the comprehension of functional requirements. ACM Trans. Softw. Eng. Methodol. 24, 1, Article 1 (September 2014), 38 pages.
DOI: http://dx.doi.org/10.1145/2629457

1. INTRODUCTION

It is widely recognized that a substantial portion of software defects (up to 85%) originates in the requirements engineering phase of the software development process [Young 2001]. Defects originated in this phase are typically caused by ambiguous, incomplete, inconsistent, silent (unexpressed), unusable, over-specific, and verbose

Author's addresses: F. Ricca, G. Reggio, and E. Astesiano, Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS), University of Genova, Vai Balbi, 5-16126 Genova, Italy; email: Filippo.Ricca@unige.it; {Gianna.Reggio; astes}@disi.unige.it; G. Scanniello, Department of Mathematics, Computer Science and Economics, University of Basilicata, Via Nazario Sauro, 85-85100, Potenza, Italy; email: giuseppe.scanniello@unibas.it; M. Torchiano, Department of Control and Computer Engineering, Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Torino, Italy; email: marco.torchiano@polito.it.

DOI: http://dx.doi.org/10.1145/2629457

requirements (both functional and nonfunctional). Defects might also stem from communication problems among stakeholders [Meyer 1985].

To face these issues, a number of methods/techniques have been proposed for representing functional and nonfunctional requirements. With regard to functional requirements, use case narratives (simply use cases hereafter) are widely employed to specify the purpose of a software system and to produce its description in terms of interactions between actors and that system [Bruegge and Dutoit 2003; Cockburn 2000]. Actors are entities (e.g., a physical system, a human being, or another software system) that interact with the subject system and that are external to it. Use cases focus on the system behavior and describe functionalities that yield results for actors. They are written in natural language (according to a more or less rigorous template) so: (i) promoting the communication among stakeholders and (ii) enabling stakeholders also with little or no design and modeling experience to comprehend functional requirements [Cockburn 2000].

Screen mockups (also called user interface sketches or user interface mockups) are used for prototyping the user interface of a subject system [Hartson and Smith 1991; van der Linden et al. 2003; O'Docherty 2005]. Mockups can be used in conjunction with use cases to improve the comprehension of functional requirements and to achieve a shared understanding on them. Research work has been done on the benefits deriving from use cases (e.g., Andrew and Drew [2009] and Anda et al. [2001]), while the effect of screen mockups has been investigated neither when they are used alone nor when they are used in conjunction with other notations. Such investigations are relevant because they would increase our body of knowledge on how to reduce software defects originating in the requirements engineering phase, and on how to improve communication among stakeholders (e.g., users, clients, and developers).

To assess whether stakeholders do actually benefit from the presence of screen mockups in the comprehension of functional requirements represented with use cases, we have conducted a family[1] of four controlled experiments with students having different profiles (e.g., Computer Science Bachelor's degree students and Computer Engineering Master's degree students). The goal of each experiment in our family of experiment is threefold: (1) evaluating the screen mockups effect on the comprehension of functional requirements, (2) investigating whether the use of screen mockups impacts the time needed to accomplish a comprehension task, and (3) understanding whether the use of screen mockups improves efficiency in performing requirements comprehension tasks (computed as the ratio between reached comprehension level and task completion time).

In this article, we present our investigation observing the following structure. First, we summarize the background on use cases and screen mockups (Section 2). Then, we provide the definition of the experiments, including the analysis methodology and the foreseen threats to validity (Section 3). After that, we present the results from the analysis of the experimental data (Section 4) and we discuss the results of our study and some implications for researchers and practitioners (Section 5). Finally, we discuss related work (Section 6) and conclude this article with final remarks and future directions for our research (Section 7).

2. USE CASES AND SCREEN MOCKUPS: A PRIMER

2.1. Use Cases

Several templates are available in the literature to specify use cases (e.g., Cockburn [2000], Jacobson [2004], and Adolph et al. [2002]). Some of them are very simple and suggest to describe uses cases as plain text (resembling Extreme Programming

[1]A family of experiments is composed of multiple similar experiments that pursue the same goal in order to build the knowledge that is needed to extract and to abstract significant conclusions [Basili et al. 1999].

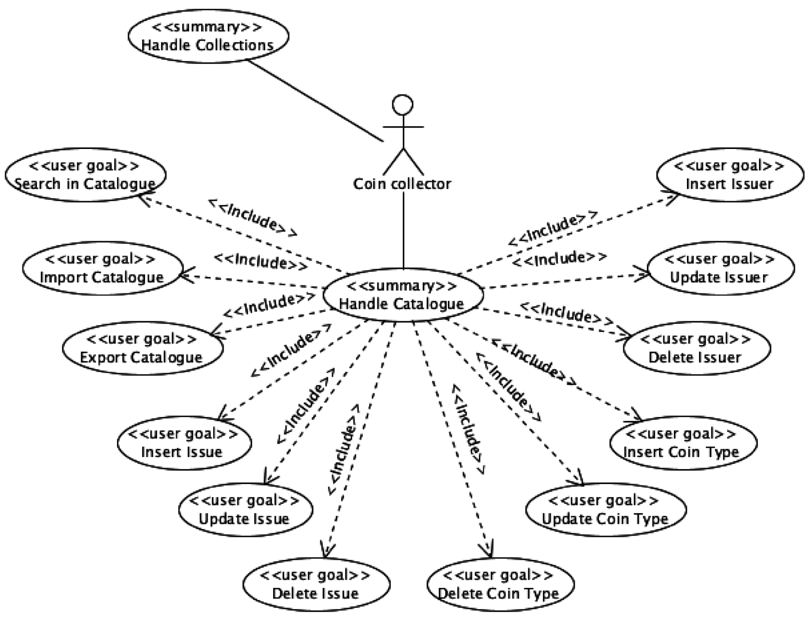


Fig. 1. EasyCoin use case diagram fragment.

user stories) [Adolph et al. 2002]. Other templates are more formal [Fowler 2003]. They all represent a use case as a main success scenario split into a list of steps, typically defining interactions between actors and the subject system. In our family of experiments, we opted for a formal template because it has been observed that the use of this kind of templates reduces ambiguities in use cases [Yue et al. 2009]. The template considered in our family of experiments is based on the SWEED method.[2]

In that template, use cases are classified with respect to their granularity level: summary (representing a goal of the system), user-goal (representing functionalities from the user perspective) and subfunction (moving out an isolated part of a scenario to a separate use case). A summary use case might include user-goal use cases, which in turn might include subfunction use cases. The granularity of a use case is depicted in the use case graphical symbols (i.e., ellipses) of the UML use case diagram. Figure 1 shows an excerpt of one of the use case diagrams of EasyCoin. It is an application for coin collector support. We used it in our family of experiments. The diagram includes use cases at summary and user-goal granularity levels, as the stereotypes <<summary>> and <<user goal>> show.

The used template also requires to specify the list of actors that interact with each use case (primary actors) or that realize it (secondary actors). For each use case, it is also needed to indicate: the rationale for its specification (*Intention in context*), the precondition/s to be verified before its execution, and the postcondition/s to be verified when its execution concludes.

Finally, the main success scenario of the use case is presented as a sequence of steps. Alternative scenarios (i.e., deviation from the main scenario) are shown as extensions of the main scenario. A step may be:

—a sentence to describe an interaction of an actor with the system or vice-versa;
—the inclusion of another use case;
—the indication of the repetition of a group of steps.

[2]www.uml.org.cn/requirementproject/pdf/re-a2-theory.pdf.

Use case: Handle Catalogue
Level: Summary
Primary actor: Coin collector
Intention in context: The Coin collector wants to create and maintain a coin catalogue (**6) suitable to her/his interests
Main success scenario:
1. The Coin collector selects the mode "Easy Catalogue".
Repeat any number of times: a step selected among 2,. . . , 13
2. The Coin collector inserts an issuer ("Insert Issuer").
3. The Coin collector updates an issuer ("Update Issuer").
4. The Coin collector deletes an issuer ("Delete Issuer").
5. The Coin collector inserts a coin type ("Insert Coin Type").
6. The Coin collector updates a coin type ("Update Coin Type").
7. The Coin collector deletes a coin type ("Delete Coin Type").
8. The Coin collector inserts an issue ("Insert Issue").
9. The Coin collector updates an issue ("Update Issue").
10. The Coin collector deletes an issue ("Delete Issue").
11. The Coin collector executes a search in the catalogue ("Search Catalogue").
12. The Coin collector imports a catalogue ("Import Catalogue").
13. The Coin collector exports the catalogue ("Export Catalogue").
14. The use case ends with success.

Fig. 2. "Handle Catalogue" Use Case. Sans-serif terms represent included use cases. (**6) refers to item 6 of the glossary.

The main success scenario and the alternative ones end with conditions indicating success or failure. This makes clear in which case the functionality is correctly or incorrectly accomplished.

References to a glossary of the terms[3] can be added to the steps of both main success scenario and alternatives ones as well as to *Intention in context*. The main objective of a glossary is to: (i) shorten use cases; (ii) reduce ambiguities (e.g., to avoid using different ways to refer to the same entity); and (iii) clarify the meaning of steps of a scenario (e.g., a richer description of the various entities could be given).

Figure 2 and Figure 3 report two use cases of the system EasyCoin. The use case in Figure 2 is a summary level use case. It specifies that one of the goals of EasyCoin is to allow *Coin collector* to build his/her own coin catalogue. *Intention in content* includes a reference to the glossary (i.e., **6), where the structure of a coin catalogue is specified (not shown here). The sentence "Repeat any number of times: a step selected among 2., . . . , 13." specifies that the steps from 2 to 13 can be repeated. Many steps of this use case are inclusions of other use cases, that is, 2 to 13. The included use cases are related by the inclusion relationships to Handle Catalogue in the use case diagram of Figure 1. On the other hand, the use case in Figure 3 describes the functionality that allows *Coin collector* (i.e., the primary actor) to insert a new type of coin. It is constituted by a main success scenario (when the use case ends successfully) and one extension (when the *Coin collector* does not confirm the insertion, and thus the use case fails). The information characterizing a coin type (e.g., material, weight, and diameter) is presented by the item 2 of glossary (i.e., **2). In this use case, it is clearly defined which information *Coin collector* has to insert about a new coin type.

2.2. Screen Mockups

Mockups are drawings that show how the user interface of a subject software system is supposed to look during the interaction between that system and the end-user (user-system interaction). Mockups may be very simple, just to help the presentation

[3]www.ibm.com/developerworks/rational/library/4034.html.

Use case: Insert Coin Type
Level: User-goal
Primary actor: Coin collector
Intention in context: The Coin collector wants to insert a new type of coin in the Catalogue
Precondition: A non-empty list of issuers of coins is displayed
Main success scenario:
1. The Coin collector selects an issuer of coins from the provided list and asks to insert a new coin type.
2. The System shows all the coin types for the selected issuer and asks the Coin collector to insert information (** 2) for the new coin type (insertCoinTypeMockup).
3. The Coin collector inserts the information.
4. The System asks the Coin collector to confirm the information added
5. The Coin collector confirms
6. The System informs the Coin collector that the insertion was successful (showCoinInformationMockup). The use case ends successfully
Extensions:
5.a The Collector does not confirm the operation. The use case fails.

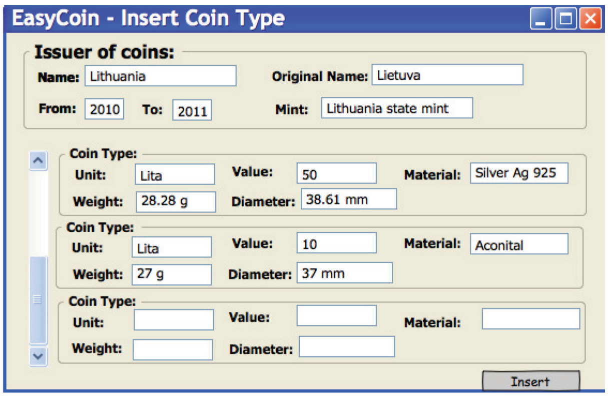
Fig. 3. "Insert Coin Type" Use Case. Underlined terms represent hyperlinks to screen mockups. (**2) refers to item 2 of the glossary.

Table I. Mockup Tools

Tool	URL
Balsamiq Mockups	`http://www.balsamiq.com/products/mockups`
Pencil Project	`http://pencil.evolus.vn/`
OverSite	`http://taubler.com/oversite/`
GUI Design Studio	`http://www.carettasoftware.com/guidesignstudio/`
Axure	`http://www.axure.com/`
Moqups	`https://moqups.com`
MockFlow	`http://www.mockflow.com/`
Mockingbird	`https://gomockingbird.com/`
SketchFlow	`http://www.microsoft.com/silverlight/sketchflow/`
iPlotz	`http://iplotz.com`

of the user-system interactions, or more detailed with rich graphics, whenever specific constraints on the graphical user interface are highlighted (e.g., requiring to use specific logos or brand related colors) [O'Docherty 2005].

Although a number of tools exists for drawing screen mockups (Table I reports a list of popular mockup drawing tools), several professionals prefer to sketch screen mockups on paper. This kind of approach has some drawbacks, which are mainly related to the continuous evolution of functional requirements [Landay and Myers 1995]. Mockups created with computer drawing tool mitigate this concern, so making them a viable alternative. Screen mockups could be also built using a programming language (e.g., Java) or an integrated development environment (e.g., NetBeans). This choice has the benefit to reuse the source code of screen mockups later in the development phase, while the main drawbacks concern the effort and the skills needed to build and maintain these mockups: any kinds of stakeholders (even without design and development experience) should be able to build mockups spending a few minutes [Ricca et al. 2010c; Scanniello et al. 2013]. In fact, one of the main barriers to the adoption of modelling techniques is the lack of competence [Torchiano et al. 2013], so a simple solution requiring no special skills has the capability of overcoming such hurdle. The use of special conceived tools


EasyCoin - Insert Coin Type

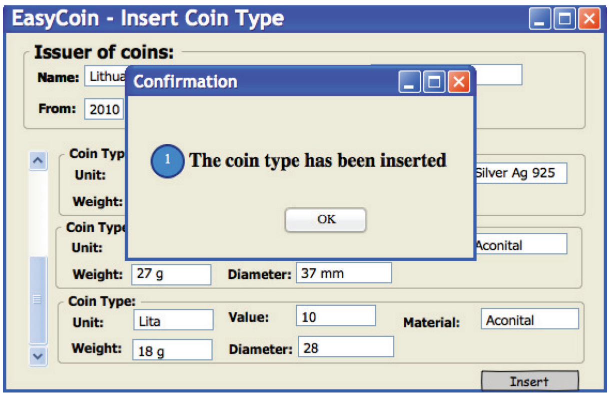
Issuer of coins:
Name: Lithuania
Original Name: Lietuva
From: 2010
To: 2011
Mint: Lithuania state mint

Coin Type:
Unit: Lita
Value: 50
Material: Silver Ag 925
Weight: 28.28 g
Diameter: 38.61 mm

Coin Type:
Unit: Lita
Value: 10
Material: Aconital
Weight: 27 g
Diameter: 37 mm

Coin Type:
Unit:
Value:
Material:
Weight:
Diameter:

(a) InsertCoinType screen mockup


EasyCoin - Insert Coin Type

Issuer of coins:
Name: Lithua
From: 2010

Confirmation
The coin type has been inserted

Coin Typ
Unit:
Weight:
Silver Ag 925

Coin Type
Unit:
Aconital
Weight: 27 g
Diameter: 37 mm

Coin Type:
Unit: Lita
Value: 10
Material: Aconital
Weight: 18 g
Diameter: 28

(b) ShowCoinInformation screen mockup

Fig. 4. Examples of screen mockups.

for drawing screen mockups represents a viable tradeoff between sketching screen mockups on paper and implementing them with a programming language. Figure 4(a) and Figure 4(b) show two screen mockups for the EasyCoin application created by the Pencil tool.[4] All the mockups in our experiments have been created by using this tool.

2.3. Augmenting Use Cases with Mockups

Even using a formal template and a glossary of terms in the definition of use cases, the requirements could still remain ambiguous, unusable, and/or difficult to comprehend. These issues can stem from communication problems among stakeholders [Meyer 1985] and consequently may result in problems (e.g., the requirements do not represent the client's view and there are functional or nonfunctional requirements that contradict each other) in the software system under development. To avoid these issues, use cases are often complemented with *screen mockups* [O'Docherty 2005]. For example, the results of a study by Ferreira et al. [2007] confirm that screen mockups can significantly improve the quality of the relationship between user interface designers and software developers.

[4]http://pencil.evolus.vn.

Screen mockups are associated with the steps of the main success scenario of a use case and possibly to its extensions that involve human actors. Indeed, mockups should be associated with the steps considered more relevant (it is the requirements analyst who makes this decision) to show what the human actor will see at that moment, when he/she will interact with the software system under development. In more detail, let M be a mockup associated with a step S. If the subject of S is the system and it has to communicate some information to an actor, then the role of the mockup M is to visualize that information (see, e.g., Figure 4(b)). If the subject of S is a primary actor, then the mockup should include the widgets (e.g., button, menu, check box) to allow the actor to enter the information needed to complete the step S (see, e.g., Figure 4(a)). The requirements analyst should check that any information displayed in a screen mockup associated with a step of a use case is either mentioned in the use case itself or in other use cases that include it.

The two screen mockups associated with the use case Insert Coin Type (Figure 3) are those shown in Figures 4(a) and 4(b), respectively. The association of screen mockups to a use case is represented in the steps of the main success scenario and of the extensions: underlined terms represent links to screen mockups (steps 2 and 6). The screen mockup in Figure 4(a) shows the selected issuer of coins (i.e., Lithuania) and the list of coin types associated with it (i.e., Lita 50 and Lita 10). On the bottom of that mockup, there are the fields the Coin collector must fill in to add a new coin type in the catalogue. On the other hand, Figure 4(b) shows the situation after the Coin collector has correctly inserted all the required information and confirmed the insertion: the system informs that the insertion was successful. Further information on how screen mockups can be built and linked to use cases can be found in Astesiano and Reggio [2012].

3. EXPERIMENTS DEFINITION AND PLANNING

On the basis of the Goal Question Metric (GQM) template [Basili et al. 1994], the goal of our family of experiments can be defined as follows.

> *Analyze* the use of screen mockups for the purpose of understanding their utility with respect to the effectiveness in comprehending requirements and the effort and the efficiency in performing comprehension tasks from the point of view of the requirements analyst, developer, and customer in the context of students reading two requirements specification documents for desktop applications.

Requirements analysts could adopt mockups as a functional requirement specification tool, while *developers* could take advantage of the improved comprehensibility of functional requirements obtained from the use of mockups. Finally, *customers* could see mockups as a viable means to validate functional requirements.

In our family of experiments, we formulated three main research questions corresponding to three general features.

RQ1. Does the requirements comprehension *effectiveness* vary when use cases are provided in conjunction with screen mockups?

RQ2. Does the *effort*[5] required to complete a comprehension task vary when use cases are provided in conjunction with screen mockups?

RQ3. Does the *efficiency* in performing a comprehension task vary when use cases are provided in conjunction with screen mockups?

[5]We consider the time as an approximation for effort. Other aspects that may be related to effort, for example, the cognitive effort of the participants, were not measured. This is customary in literature and it is compliant with the ISO/IEC 9126 standard [ISO 1991] definition: effort is the productive time associated with a specific project task.

Table II. Overview of the Experiments

Main factor:	Requirements specification documents: - With screen mockups (**S**) **or** - Without screen mockups (**T**)
Systems Used:	- EasyCoin is a system for cataloguing collections of coins - AMICO is a system for the management of condominiums
Tasks:	The participants sequentially performed two comprehensions tasks on the systems EasyCoin and AMICO
Participants/Profiles:	- Undergraduate Computer science students with two levels of experience; - Graduate students (Computer Engineering) - Graduate students without modeling and development experience (Maths and Telecommunication)
Dependent Variables:	Comprehension level, task completion time, and task efficiency
Perspective:	Requirements analyst, developer, and customer

Table III. Details of Experiments

	UniBas1	**UniGe**	**PoliTo**	**UniBas2**
Date	Nov. 2009	Dec. 2009	Apr. 2010	Dec. 2010
Location	Univ. Basilicata	Univ. Genova	Poly. Torino	Univ. Basilicata
Degree	Computer Science	Computer Science	Computer Engineering	Math/Telecom
Level	Undergrad	Undergrad	Graduate	Graduate
Year	2nd	3rd	2nd	1st
Semester	1st	1st	2nd	1st
Number	33	51	24	31

The essential common characteristics of our family of experiments are summarized in Table II. The experiments exhibit a few differences in terms of context, namely the number and characteristics of the involved participants and where the experiments have been executed. Table III reposts such information, together with the labels of the experiments (e.g., UniBas1), and when each experiment has been executed (i.e., month, year, and semester). The number of participants for each experiment is shown as well (e.g., 33 for UniBas1).

We conducted the first experiment at the University of Basilicata (UniBas1). The other experiments in our family (UniGe, PoliTo, and UniBas2) are *differentiated* replications [Basili et al. 1999]: the same experiment design and procedure were used, but different kinds of participants were involved. On the other hand, these replications can also be considered *dependent* (the experiment conditions are similar) and *exact* (the followed experiment procedure was similar in the experiments) [Shull et al. 2008]. These replications can be also classified as *internal* because they were conducted by the same group of researchers as the original experiment [Mendonça et al. 2008].

3.1. Experimental Objects

We selected as objects for the experiments the requirement specification documents of two desktop applications.

Table IV. Experiment Design. S is for requirements specification with screen mockups and T without them

	Group 1	Group 2	Group 3	Group 4
First laboratory run	S, EasyCoin	T, EasyCoin	T, AMICO	S, AMICO
Second laboratory run	T, AMICO	S, AMICO	S, EasyCoin	T, EasyCoin

EasyCoin is an application for coin collector support (used as running example in Section 2);
AMICO is a software for condominium management.

Depending on the treatment, use cases within these specification documents were augmented (or not) with screen mockups. These documents were small enough to fit the time constraints of the experiments though realistic for small-sized development projects of the following kinds: in-house software (the system is developed inside the software company for its own use), product (a system is developed and marketed by a company, and sold to other companies), or contract (a supplier company develops and delivers a system to a customer company) [Lauesen 2002].

The complexity of the requirement specification documents for the two systems can be considered comparable. The specification documents of AMICO and EasyCoin contain 19 and 20 use cases, respectively. The number of screen mockups is 31 for AMICO and 32 for EasyCoin. The number of pages is 25 for both documents (using a 12 pt font when the screen mockups are present).

In the specification documents, screen mockups (when present) were used only for use cases deserving more attention (e.g., when the interaction between the actors and the systems could be a source of misunderstanding). Each use case could be enhanced with more than one screen mockup (e.g., this is the case in Figure 3) and references to a given mockup could appear in more than one use case. Mockups have varying size and complexity. Finally, these mockups did not provide any additional information that could not be derived from use cases and glossary.

The specification documents of AMICO and EasyCoin were realized by one of the authors in two editions (2006 and 2008) of the Software Engineering course at the University of Genova [Astesiano et al. 2007] and used within the laboratory part of that course to design and implement the corresponding systems. The specification documents underwent a number of modifications since their first version (11 versions for EasyCoin and 12 for AMICO) until the execution of the family of experiments presented here. The execution of modification operations are common in software indystry to improve the overall quality of such a kind of specification documents. We can consider the overall quality of the used documents comparable with each other.

The documents (in Italian language) can be found, together with all the experimental materials, in the experimental package available on the web (www2.unibas.it/gscanniello/ScreenMockupExp).

3.2. Design

All the experiments in our family were conducted using the same design. We selected a *within-participants counterbalanced design* [Wohlin et al. 2012] following the recommendations provided by Wohlin et al. [2012], Juristo and Moreno [2001], and Kitchenham et al. [2002]. The used design is schematically shown in Table IV. Each trial or experimental unit/run consists of a task to be performed on an object (either AMICO or EasyCoin requirements specification documents) that may have been exposed to screen mockup (S) or not (T). This design ensures that the participants in each experiment work on different experiment objects in two subsequent laboratory runs (first run and second run), with screen mockups available in only one of the two

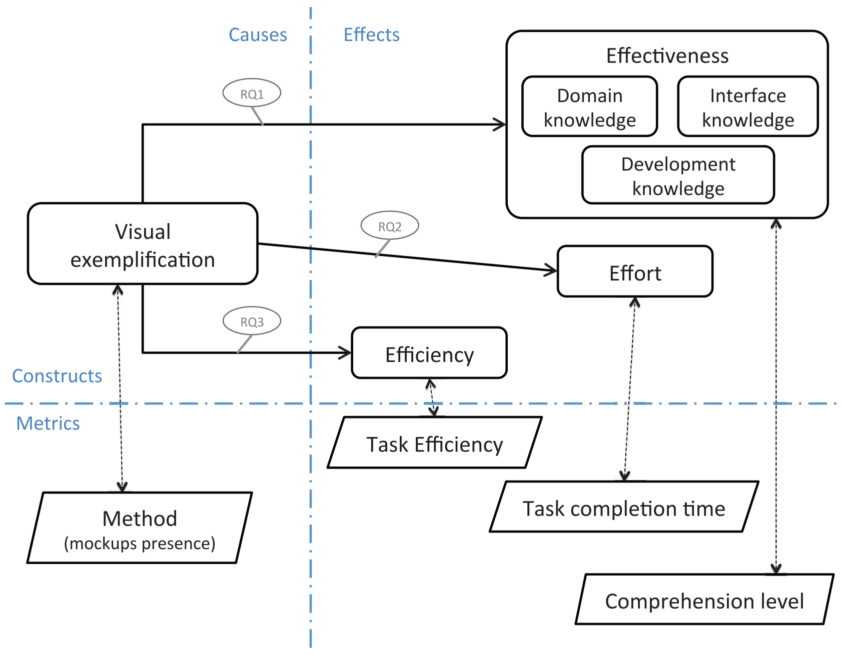


Fig. 5. Conceptual model.

runs. For instance, participants in Group 1 accomplished in the first laboratory run a comprehension task on EasyCoin and the specification document included uses cases augmented with screen mockups, while they worked in their second run on AMICO and the specification document did not contain screen mockups.

A break half-hour between the two laboratory runs was allowed. Moreover, the experimental material needed for the second laboratory run was provided only when all the material of the first run was given back. We used the GPA[6] (Grade Point Average) of the participants to split them into the four groups (i.e., Group 1, Group 2, Group 3, and Group 4). We equally distributed (as much as possible) high- and low-ability participants among these groups. As suggested by Abrahão et al. [2013], a student with a GPA less than or equal to 24 can be considered to be a low-ability participant, otherwise high. Participants' ability represents the blocking factor for the experiments in our family of controlled experiments.

3.3. Variable Selection and Hypotheses Formulation

To quantitatively investigate the research questions delineated previously, we first ought to make our conceptual model—that is, the abstract constructs involved in our research questions—explicit. Our basic conceptual model is reported in the upper part of Figure 5. Building upon our own experience and an analysis of the literature about use cases and screen mockups, we could postulate that *Visual exemplification*—screen mockups in particular—has an effect on both the effectiveness of functional requirements comprehension, the effort required to acquire such a comprehension, and ultimately the efficiency of the task. Since comprehension encompasses the acquisition of

[6]In Italy, the exam grades are expressed as integers and assume values between 18 and 30. The lowest grade is 18, while the highest is 30.

different kinds of knowledge [Aranda et al. 2007], we focused on domain knowledge, interface knowledge, and implementation knowledge.

The relationship between the conceptual model and our research questions is shown in Figure 5. RQs are reported as small ellipses and are linked to the effects they address. The bottom part of Figure 5 reports the metrics we defined to measure the effect.

The main factor of the experiments in our family consists in the presence or absence of Visual exemplification (i.e., screen mockups) in the requirements specification document. It is as follows.

Method. This variable (also called "main factor" from here on) indicates whether requirements are used in conjunction with screen mockups or not. It is a nominal variable that can assume two values: S (requirements specifications with screen mockups) or T (requirements specifications without screen mockups). We considered participants who received the specification documents without screen mockups as the *control group*, while the *treatment group* was comprised of participants who received the specification documents with the screen mockups.

RQ1 concerns the effect of screen mockups on the Effectiveness of requirements comprehension. The instrument we devised to measure the Comprehension construct is a *comprehension questionnaire*. We designed two questionnaires, one for each system. The same questionnaire was used for a given system independently from S and T. Each questionnaire is composed of a fixed number of items (i.e., questions)—that is, ten for both AMICO and EasyCoin in all the experiments in our family, with the exception of UniBas2 where the administered items were seven (further details are provided following this section)—formulated so as to require an open response. More precisely, each questionnaire item required an answer consisting of a list of one or more elements. We consider an item response to be correct if all the affected or relevant elements are mentioned and no unaffected or irrelevant elements are mentioned. The same approach was used by Kamsties et al. [2003]. The Comprehension construct is measured by the following dependent variable.

Comprehension Level. It measures the comprehension a participant achieved on the functional requirements of a system. We computed the comprehension level as the number of correct answers divided by the number of items in the comprehension questionnaire. Comprehension level is a ratio metric that assumes values in the interval [0, 1].

A value close to 1 means that a participant got a very good comprehension of functional requirements because he/she correctly answered all the items of a comprehension questionnaire, while a value close to 0 indicates no or bad comprehension. A sample question (i.e., Q4) for the EasyCoin questionnaire (the complete questionnaire is reported in Table XIV of Appendix A) is: "What has the Collector to select before inserting a coin type? What is a coin type linked to?". An answer to Q4 is considered correct if and only if "*issuer of coins*" is given and no other extraneous element/datum is reported by participants. The correct answer can be deduced from: (i) the point 1 of the main success scenario of the use case shown in Figure 3 or (ii) the screen mockup shown in Figure 4(a). To answer that question, the information provided by the use case and the screen mockup is the same, but presented in a different way. Therefore, in the comprehension of functional requirements stakeholders with different experience and background might have different benefits in using: (i) use cases alone or (ii) use cases augmented with screen mockups. This explains why we decided to conduct replications with different profiles of participants. An example of comprehension question for the AMICO system (Q1) is: "List the atomic data associated with a given condominium" (for the complete questionnaire, see Table XIII in Appendix A). The answer is considered correct if all the atomic elements/data are reported (i.e., identifier, address, post code, state) and no other extraneous element/datum is reported.

To quantitatively measure comprehension, we opted for the measure above because it is very simple, at least compared to a more fine-grained approaches used for similar purposes (e.g., the one based on the information retrieval metrics precision and recall adopted by Abrahão et al. [2013] and by Ricca et al. [2010a]). In addition, the advantages of that choice should be: ease of understanding for the readers and more conservative conclusions.

Since the main Comprehension construct is decomposed into subconstructs corresponding to three knowledge areas, we grouped the items of the AMICO and EasyCoin comprehension questionnaires into three corresponding groups. The first group includes three items and regards aspects related to problem domain (simply *domain* in the following), the second group includes four items and concerns interactions between actors and system, and execution flow (*interface* in the following). The last three items regard aspects related to solution domain (*development* in the following). Therefore, we studied also the comprehension level in each knowledge domain in addition to the overall comprehension level on the requirements. In practice, the correctness of the answers for each group was computed as for the variable comprehension level, but considering only the relative items. As a consequence, the overall comprehension level (CL) for a participant can be expressed as a linear combination of the knowledge levels for domain (CL_{Domain}), interface ($CL_{Interface}$), and development ($CL_{Development}$):

$$CL = \frac{3}{10}CL_{Domain} + \frac{4}{10}CL_{Interface} + \frac{3}{10}CL_{Development}.$$

Research question RQ2 pertains to the influence of Visual exemplification on the Effort required to comprehend requirements. The Effort construct is measured by means of the following variable.

Task Completion Time. It indicates the minutes spent by a participant to complete the tasks defined in the comprehension questionnaire described previously. The time was recorded directly by the participants on the questionnaire, as begin- and end-time both expressed in terms of hours and minutes. The completion time is a ratio variable that can assume positive integer values only.

The unit of measurement was selected to achieve a reasonable accuracy and ease of reporting. We opted for measuring the cumulative time for filling in the full questionnaire instead of the time for each item for two reasons: first, to avoid disrupting too much the work, and second, to minimize the relative error since we expected the time for each question to be in the order of a few minutes, which is the granularity of the measure. We opted for a very simple measurement of the Effort construct: we are aware that time cannot completely capture the cognitive effort of the participants when performing the tasks. Actually, we were more concerned with the cost aspect of the effort, which a project manager would equate to time by applying a pragmatic simplification.

Research question RQ3 concerns the influence of Visual exemplification on the Efficiency with which a comprehension task is accomplished. The Efficiency construct is measured by means of the following variable.

Task Efficiency. It measures the efficiency of a participant in the execution of a comprehension task. The efficiency is a derived measure that is computed as the ratio between comprehension level and task completion time. To provide an easy-to-understand meaning, we decided to measure the task efficiency measures in *comprehension tasks* *per hour*, with an answer to an item in the comprehension questionnaire considered a—basic—comprehension task. Considering that the task time is measured in minutes the efficiency is defined as:

$$TaskEfficiency = \frac{CL \cdot N_{items}}{tt/60}, \tag{1}$$

where: CL is the overall comprehension level achieved by the participant, N_{items} is the number of answered items in the comprehension questionnaire, and tt is the time measured in minutes employed by the participant to complete the task.

Task efficiency is a ratio measure that ranges in between 0 and 600. The largest value indicates that a hypothetical participant in one minute ($tt = 1$ the smallest unit of time) achieves a perfect comprehension ($CL = 1$) of all the items ($N_{items} = 10$). The perspective we adopted for measuring efficiency is that of the quality in use (see, e.g., ISO 9241-11 [ISO 2000], ISO/IEC 25010 [ISO 2011]): efficiency measures the effectiveness achieved to the expenditure of resources. The peculiarity of our study is that we adopt an approach, which was originally defined to evaluate the use of a piece of software, to evaluate a specific documentation: we practice the principle that software documentation is software, too.

To answer our research questions, we defined the following null hypotheses.

H_{c0}. The presence of screen mockups *does not significantly improve* the comprehension level of functional requirements.
H_{t0}. The presence of screen mockups *does not significantly affect* the time to accomplish a comprehension task.
H_{e0}. The presence of screen mockups *does not significantly affect* the efficiency of the comprehension task.

The first hypothesis is one-tailed because we expect a positive effect of screen mockups on the comprehension of functional requirements. While we expect some effect on Task completion time, we have no theory or previous evidence that points in one specific direction, thus the second null hypothesis is two-tailed. Indeed, even though it can be postulated that the participants in the treatment group were provided with additional information, it could also be the case that the extra information required more time to complete a comprehension task. Our postulation is supported by the framework by Aranda et al. [2007], which is based both on the underlying theory of the modeling language and on the cognitive science. For similar reasons, H_{e0} is two-tailed. These hypotheses were tested within each experiment, then the results were considered together and commented.

These hypotheses address the macroscopic effect of screen mockups on comprehension. The essential reason why we expect to observe some effect is that we believe screen mockups represent an important source of information and they will be used effectively to comprehend functional requirements. That is, source of information allows us to get some indications from a qualitative perspective on how the participants used the representations given to deal with comprehension tasks. In particular, the possible source of information we identified are

—glossary,
—previous knowledge,
—use cases,
—use case diagram,
—screen mockups.

Table V. Summary of Variables Used in the Study

Variable	Type	Description	Scale
Method	Indep.	whether requirements are augmented with screen mockups	Nominal $\in \{S, T\}$
Application	Indep.	experimental object used in task	Nominal $\in \{AMICO, EasyCoin\}$
Lab	Indep.	order of the experiment unit within the experiment for the participant	Ordinal $\in \{1, 2\}$
Experiment	Indep.	participants' profiles and experiment	Nominal $\in \{UniBas1, UniGe, PoliTo, UniBas2\}$
Comprehension level	Dep.	comprehension achieved by a participant on the functional requirements	Ratio $\in [0, 1]$
Task completion time	Dep.	time spent by a participant to complete a comprehension task	Interval $\in (0, \infty)$
Task efficiency	Dep.	task time efficiency	Ratio $\in [0, 600]$
Source	Dep.	(main) source of information used to perform comprehension task	Nominal $\in \{G, K, UC, UCD, S\}$

As suggested by Aranda et al. [2007], we analyzed the (main) source of information the participants used to answer each item in the comprehension questionnaire of both the applications used in our family of experiments. The participant was asked to specify only one source of information for each item in the comprehension questionnaire. The following variable has been considered.

Source. It is a nominal variable that can assume the values: Glossary (G), previous Knowledge (K), Screen mockups (S), Use Cases (UC), or Use Case Diagrams (UCD).

In our family of experiments, we also analyzed the effect of the participants' profiles on the comprehension of functional requirements.

Experiment. It indicates the experiment in our family. Therefore, Experiment is a nominal variable that can assume the following values: UniBas1, UniGe, PoliTo, and UniBas2.

We also analyzed possible co-factors, whose effect could be confounded with the main factor (i.e., the manipulated factor).

Application. It indicates which experiment object is used in the trial. It is a nominal variable that can assume two possible values: AMICO or EasyCoin.

Lab. It indicates in which experiment run the task was conducted (first or second). Due to the experiment design, it is an ordinal variable with two levels.

In Table V, we summarize the independent and dependent variables used in our family of experiments.

3.4. Procedure

The participants accomplished comprehension tasks using computers equipped with MS Word. An Internet connection was available, while performing the comprehension tasks. We provided the participants with the following material.

Table VI. Postexperiment Survey Questionnaire

Item ID	Question	Valid Answers
PQ1	I had enough time to perform the tasks.	(1-5)
PQ2	The questions of the comprehension questionnaire were clear to me.	(1-5)
PQ3	I did not have any issue in comprehending the use cases.	(1-5)
PQ4	I did not have any issue in comprehending the use case diagrams.	(1-5)
PQ5	I found the exercise useful.	(1-5)
PQ6	I found screen mockups useful (when present)	(1-5)
PQ7	To see the screen mockups (when present), I spent (in terms of percentage) with respect to the total time to accomplish the task.	(A-E)
(1) strongly agree, **(2)** agree, **(3)** neither agree nor disagree, **(4)** disagree, **(5)** strongly disagree.		
(A) $< 20\%$, **(B)** $> 20\%$ and $\leq 40\%$, **(C)** $> 40\%$ and $\leq 60\%$, **(D)** $> 60\%$ and $\leq 80\%$, **(E)** $\geq 80\%$		

—The requirements specification documents in electronic format (MS Word) of EasyCoin and AMICO. In particular, each document contained
 (i) the system mission, namely, a textual description of both the functionality of the future system and the environment in which it will be deployed;
 (ii) a UML use case diagram summarizing the use cases of the systems;
 (iii) functional requirements expressed as use cases specified according to the chosen template. Depending on the experiment design, use cases were or were not complemented with screen mockups;
 (iv) a glossary of the terms.
—A paper copy of the comprehension questionnaires of EasyCoin and AMICO.
—A paper copy of the postexperiment questionnaire shown in Table VI.
—The training material, which included a set of instructional slides describing the template employed for the specification of the use cases, some examples not related with the experiment objects, and a set of slides describing the procedure to follow in the task execution.

We opted for an electronic format of the requirements specification document to permit the "Find" facility that is well known also to less-experienced participants and its use is convenient for large-sized documents. Furthermore, when mockups were present, the participants could click the hyperlinks in the use cases (see Figure 3) to visualize the mockups.

The used postexperiment questionnaire is composed of seven items (see Table VI). That questionnaire is aimed at gaining insights about the participants' behavior in the experiment and collecting information useful to better explain quantitative results. In particular, a first group of questions (PQ1 through PQ5) concerned the availability of sufficient time to complete the tasks, the clarity of the use cases, and the ability of participants to understand them. PQ6 was devoted to the perceived usefulness of screen mockups, while PQ7 aimed at understanding how much time the participants thought they spent, in percentage intervals, analyzing use cases and screen mockups. All the items, except PQ7 that is expressed in intervals of percentages, require responses according to a five-point Likert scale [Oppenheim 1992]: (1) strongly agree, (2) agree, (3) neither agree nor disagree, (4) disagree, and (5) strongly disagree.

For each task, the participants went through the following procedure.

(1) Specify name and start-time on the comprehension questionnaire.
(2) Fill in individually the questionnaire items by consulting the requirements specification document.
(3) Mark the end-time on the comprehension questionnaire.
(4) At the end of the second laboratory run, we asked each participant to fill in the postexperiment questionnaire.

We did not suggest any approach on how to tackle the comprehension tasks (e.g., read a question and use the "Find" facilities of MS Word to identify the portion of the document of interest). We only discouraged reading completely the requirements specification documents to avoid wasting time. The participants could ask clarifications on the items of the comprehension questionnaires. That is, supervisors only explained the items without providing any information on the possible answers.

3.5. Preparation

A pilot experiment was executed before the original experiment to: (i) assess experiment material and (ii) get indications on the time to execute comprehension tasks on EasyCoin and AMICO. We asked two students (one of the Bachelor program and one of the Master program in Computer Science both at the University of Genova) to execute the comprehension tasks. Also, two of the authors not involved in the preparation of the material executed the pilot experiment. For the students and the authors, the pilot completion time was, respectively, 118, 130, 62, and 53 minutes. The results of the pilot suggested that the experiment was well suited for Bachelor and Master students and that 3 hours were sufficient (break and time needed to deliver the material included). Minor changes to the comprehension questionnaires were made following the comments of the participants in the pilot experiment.

About 2 weeks before each experiment took place, the participants attended a lesson of 2 hours on use case modeling based on the SWEED method (except for UniGe, see next section). The participants performed also a training session, where they were asked to execute a comprehension task on a different system, that is, LaTazza (with the exception of UniGe, see next section). LaTazza is a coffee maker support application to manage the sale and the supply of small-bags of beverages. To get some demographic information about the participants, we also asked the participants to fill in a *prequestionnaire* (except for UniGe, see next section). The prequestionnaire was also used to capture information about the background (e.g., attended courses) and industrial experience of the participants. In addition, we used this questionnaire to get the students' GPA.

3.6. Context and participants

We conducted all the experiments in research laboratories under controlled conditions. Some differences were introduced in the replications. For each experiment, we discuss variations with respect to the original experiment and highlight the characteristics of the participants:

UniBas1. It was conducted at the University of Basilicata with 2nd-year Bachelor students in Computer Science. The experiment represented an optional educational activity of a Software Engineering course (first semester, from October 2009 to January 2010). It is important to highlight that UniBas1 students learned Requirements Engineering, UML, and use-case modeling in that course. As mandatory laboratory activity, the students were grouped in teams and allocated on projects to design software systems using the UML.

UniGe. It was carried out at the University of Genova with 3rd-year Bachelor students in Computer Science. The students were enrolled in a Software Engineering course held in the first semester (from October 2009 to January 2010). The UniGe students learned Requirements Engineering, the UML and use case modeling in that course on Software Engineering. As a mandatory activity, students were grouped in teams and allocated on projects to develop a software system using specifications based on the UML. Some of them are, or have been, industry professionals. The experiment was conducted as part of laboratory exercises carried out within the course in which the experiment was conducted. Some differences have been introduced with respect to UniBas1. They are summarized as follows.

—The participants did not attend a lesson on the SWEED method. We introduce this difference because the participants in UniGe studied this method in the Software Engineering course they have been attending.
—An exercise similar to the comprehension tasks of the experiment was not conducted before the replication for the same motivation as mentioned previously.
—A half-hour break between the two laboratory runs was not provided for time constraint (the laboratory was available for a fixed time).
—We randomly assigned the participants to the groups in Table IV. This was because of time constraints that did not allow participants to fill in a pre-questionnaire.

It is worth mentioning that in a previous paper by Ricca et al. [2010b], a preliminary analysis and some results from UniGe have been presented. The original experiment and the other replications are presented in this article for the first time. Another noteworthy difference with respect to that paper concerns the method used to quantitatively assess the correctness of the answers for the comprehension questionnaires of AMICO and EasyCoin. In our previous paper [Ricca et al. 2010b], we used the information retrieval metrics precision and recall.

PoliTo. The same design and conditions as UniBas1 were employed to conduct this replication. The participants were 2nd-year graduate students at the Polytechnic of Turin. They were enrolled in an Advanced Software Engineering course, which was held in the second semester (from March 2010 to June 2010). The experiment was conducted as a laboratory exercise in this course. The participants had a reasonable level of technical maturity and knowledge of the UML, requirements engineering, use case modeling and object-oriented software development.

UniBas2. It involved graduate students at the University of Basilicata from: (i) a Mathematics Master's degree program and (ii) a Telecommunication Master's degree program. The replication was an optional exercise of an Algorithm and Data Structure course. UniBas2 was held in the first semester, from October 2010 to January 2011. With respect to UniBas1 and the other replications, we deliberately introduced the following difference: the items of the development category (Q8, Q9, and Q10) were removed from the comprehension questionnaires of AMICO and EasyCoin. This variation was introduced because the participants did not have any experience in software development. Moreover, they had no previous experience with requirements engineering and use case modeling.

3.7. Analysis Method

We analyze the individual experiments for each research question and then we comment on all the data from all the experiments in the discussion.

(1) We analyze each research question according to the following schema:
    (a) presentation of the general descriptive statistics,

   - (b) check for the normality of the distribution of the dependent variable (Comprehension level for RQ1, Task completion time for RQ2, and Task efficiency for RQ3),
   - (c) analysis of raw main effect of Method on the dependent variable and effect size,
   - (d) (only for RQ1) analysis of both main and combined effect of Method and the Domain knowledge on the dependent variable (comprehension level),
   - (e) analysis of the main and pairwise combined effect of Method and co-factors (Application and Lab) on the dependent variables (comprehension level for RQ1, Task completion time for RQ2, and Task efficiency for RQ3).

(2) We analyze the relevance of screen mockups as a source of information.
(3) We analyze the responses to the postexperiment questionnaire.
(4) We perform a cross-experiment analysis checking for the effect of Method and the experiment on dependent variables.

Concerning the descriptive statistics (1.a), we report the number of participants, the central tendency, both mean and median, and the dispersion as standard deviation (σ). We visually show the distribution of the values by means of box plots.

For the purpose of hypothesis testing, we used nonparametric analyses because the distribution of data was nonnormal (1.b). We verified that by means of the Shapiro-Wilk W test [Shapiro and Wilk 1965] for normality, which tests the null hypothesis that the sample is drawn from a normally distributed population. We opted for this test because of its good power properties as compared to a wide range of alternative tests. It is worth mentioning that we used parametric analyses when possible, that is, when data were normally distributed.

For testing the raw main effect of Method (1.c), we chose the Wilcoxon matched pair test (in the following, just Wilcoxon test). It is the nonparametric equivalent of the t-test, to test the difference between treatment and control groups, that can be applied also for nonnormal variables. We opted for the Wilcoxon test because it is very robust and sensitive [Motulsky 2010]. We analyzed each system separately and we also discriminated them in terms of item categories/groups (e.g., domain).

Statistical tests check the presence of significant differences, but they do not provide any information about the magnitude of such a difference. Effect size is very useful since it provides an objective measure of the importance of the experimental effect. Several commonly adopted effect size indexes, for example Cohen's d and Hedges' g, are calculated using only the means and standard deviations of the two groups being compared, therefore they are extremely vulnerable to violations of normality and may not result in appropriate quantification of the effect [Hogarty and Kromrey 2001]. Since we observed nonnormal distributions, we opted for a nonparametric effect size index, the Cliff's Delta (δ) [Cliff 1993]. We judged the magnitude of the effect size by comparing it to three thresholds: negligible if $\delta < 0.147$, small if $0.147 \leq \delta < 0.33$, medium if $0.33 \leq \delta < 0.474$, and large if $0.474 \leq \delta$ [Romano et al. 2006].

Moreover, since it is difficult to relate the Cliff's Delta value to a practical meaning, we also provide the means' percentage improvement as a less robust though more intuitive and qualitative effect size indicator. Given two values (*ctrl*, *exp*), corresponding to the control and experimental group means respectively, the means' percentage improvement (*exp* with respect to *ctrl*) can be computed as $(exp - ctrl)/ctrl$.

As far as the three distinct constructs of comprehension are concerned (1.d), we considered the knowledge domain as an additional independent variable (*KD*) and performed factorial analysis by means of a permutation test [Higgins 2004]. Permutation tests extend the applicability of ANOVA models to cases of nonnormality and heteroscedasticity, which are typical to our field. The essential idea underlying permutation tests is to compare test statistics to an empirical distribution instead of an a-priori known distribution. Permutation tests have been recently applied in some experiments similar to the one reported in this article (e.g., Ceccato et al. [2014] and Scanniello et al. [2014]) also thanks to the implementations available in popular statistical software [Kabacoff 2011].

We adopted permutation tests also for analyzing the possible confounding factors, Application and Lab (1.e). For comprehension level, we performed a single permutation test to analyze the effect of both the knowledge domain and co-factors.

Alpha (α) is a threshold value used to judge whether a statistical test is statistically significant or not; α represents a probability of committing a Type-I-error [Wohlin et al. 2012]. Because alpha corresponds to a probability, it can range from 0 to 1. The most commonly used value for alpha is 0.05. It indicates 5% chance of a Type I error occurring (i.e., rejecting the null hypothesis when it is true). If the p-value of a test is equal to or less than the chosen level of alpha, it is deemed statistically significant; otherwise it is not. Since we performed three different analyses on the comprehension-level-dependent variable, we cannot use $\alpha = 0.05$; we need to compensate for repeated tests. While several techniques are available, we opted for the most conservative one, the Bonferroni correction [Dunn 1961]. In practice, the conclusion will be taken by comparing the tests p-value to a corrected significance level $\alpha_B = \alpha/n_t$, where n_t is the number of tests performed. Thus, in our case, $\alpha_B = 0.05/3 = 0.016$. Similarly, for Task completion time and Task efficiency variables, we adopt a $\alpha_B = 0.05/3 = 0.016$.

We measured the relevance of a source of information (2)—that is, Glossary (G), previous Knowledge (K), Screen mockups (S), Use Cases (UC), or Use Case Diagrams (UCD)—as the proportion of participants using it to respond to an item in the comprehension questionnaire. An average importance source can be expected to be used by N/n_s participants, where n_s is the number of alternative sources (i.e., 5 in our case) and N the number of participants. We assess the relevance of screen mockups by testing the proportion of participants using them (when present) to be greater than the average expected proportion. In this case, mockups are considered a *relevant* source of information. In addition, we considered a source of information to be *predominant* when it is the most important source of information. To test the statistical significance of these conditions, we used a proportion test [Agresti 2007].

Regarding the postexperiment survey questionnaire (3), we verified whether the participants consistently agreed with the proposed statement. The answers were based on a five-points Likert scale ranging from “strongly agree” to “strongly disagree”, which were encoded with integers from 1 to 5. To show the answers to the postexperiment survey questionnaire graphically, we adopted clustered bar charts. These are widely employed since they provide a quick visual representation to summarize data.

Finally, regarding cross-experiment analysis (4), we conducted a factorial analysis of variance, by means of nonparametric permutation tests, considering as output each of the three main dependent variables and co-factors.

Statistical analyses were performed by using the R statistical package [R Core Team 2013]. For example, we used the *effsize* package [Torchiano 2014] for computing the Cliff’s δ estimates and the *lmPerm* package [Wheeler 2010] for the permutation tests.

3.8. Threats to Validity

We discuss the threats to validity of the controlled experiments using the guidelines proposed by Wohlin et al. [2012].

3.8.1. Internal Validity. In experiments like ours that try to establish a causal relationship, the internal validity threats are relevant because they may invalidate such relationship.

—*Interaction with Selection.* Refers to selecting participants for the various groups in the study (Are the groups equivalent at the beginning of the study?). This threat has been mitigated since each group of participants worked on different experiment objects with and without screen mockups in two comprehension tasks. Further, the participants in each experiment had similar background.
—*Maturation.* Participants might learn how to improve their comprehension performances passing from the first trial to the second one. However, the adopted experimental designs should mitigate possible learning effects as well as any possible effect deriving from the experiment objects and the tasks execution order.
—*Diffusion or Imitation of Treatments.* This threat concerns the information exchanged among the participants, while performing a task and when passing from the first laboratory run to the second one. We prevented this in several ways. While accomplishing the experiment, the participants were monitored by the experiment supervisors to prevent them from communicating with each other. The supervisors also controlled that no communication among participants occurred in the transition between the two consecutive laboratory runs. Also the communication among participants in different experiments might bias the results. As an additional measure to prevent the diffusion of material among the participants, we asked back all the material.

3.8.2. External Validity. The main issue of the external validity refers to the generalizability of the results.

—*Interaction of Selection and Treatment.* The use of students as participants may affect external validity [Carver et al. 2003; Ciolkowski et al. 2004; Hannay and Jørgensen 2008]. This threat is related to the representativeness of participants as compared with professionals. We believe that the participants in UniBas1 and UniGe are not far from novice software engineers and junior programmers since they were trained on the UML and use cases and attended and passed several programming courses (e.g., procedural programming, object-oriented programming, and databases). However, many of the participants to PoliTo were, or have been, professionals in industry. All of them had an internship in industry as their final part of their Bachelor degree. Thus, the industrial experience of these students made them not very different from software professionals of small to medium software companies [Scanniello et al. 2010]. Another possible threat to external validity is that we have conducted a family of controlled experiments in an artificial laboratory setting with specific constraints. To increase our awareness on the results, we promoted independent replications (i.e., performed by other experimenters) making available a laboratory package to the web. This will allow cheap execution and design of replications [Lindvall et al. 2005]
—*Interaction of Setting and Treatment.* In our case, the size and complexity of the documents and the systems themselves may affect the validity of the results. Larger systems could excessively overload the participants, thus biasing the results of the experiments. Different empirical investigations in terms of case studies are needed to assess whether size and complexity may affect the results obtained. Finally, the use of Word files for the requirements specifications could have affected the comprehension achieved by the participants.

3.8.3. Construct Validity. Construct validity threats concern the relationship between theory and observation.

—*Interaction of Different Treatments.* We adopted a within-participants counterbalanced experiment design to mitigate this threat.

—*Confounding Constructs and Level of Construct.* The procedure used to divide the participants into groups could affect the validity of the results especially for UniGe where participants were randomly assigned to the groups in Table IV.
—*Evaluation Apprehension.* We mitigated this threat because the participants were not evaluated on the results achieved in the experiments. We added one point to the final examination mark of each student (except for PoliTo where the final assessment was: pass - not pass) independently from the values they achieved on both the dependent variables. The participants were not aware of the experiment hypotheses.
—*Experimenters' Expectations*. We formulated the questions of the comprehension questionnaires to favor neither S nor T. We designed the postexperiment survey questionnaire to capture the participants' perception on the experiments and used standard approaches and scales [Oppenheim 1992].

3.8.4. Conclusion Validity. Conclusion validity concerns issues that may affect the ability of drawing a correct conclusion.

—*Reliability of Measures.* This threat is related to how the dependent variables were measured. Comprehension was measured using questionnaires and items responses were compared to previously determined correct ones. We also exploited an information-retrieval-based approach to quantitatively assess the answers the participants provided to the comprehension questionnaires. In particular, we used the same approach as in Ricca et al. [2010b] and performed the same analyses as we presented in this article. The achieved results confirmed the findings we have discussed here (see the web page of our experiment for details). Regarding task completion time, it was measured by means of proper time sheets and validated by researchers, who were present during the experiment. Even if this method of measuring the task completion time can be considered questionable [Hochstein et al. 2005], this practice is common in the empirical software engineering field (see, for instance, Huang and Holcombe [2009], Humphrey [1995], Ceccato et al. [2014], and Scanniello et al. [2014]).
—*Random Heterogeneity of Participants.* Regarding the selection of the population, we drew fair samples and conducted our experiments with participants belonging to these samples. Replications are however required to confirm or contradict the results we achieved.
—*Fishing and the Error Rate.* The hypotheses have been always accepted considering proper p-values. In particular, since we computed repeated tests on the dependent variables, we had to apply compensation. To execute this task, we opted for the Bonferroni correction. Finally, the sample size is large enough to give strength to the achieved results.
—*Statistical Tests.* We planned to use statistical tests that were well known for their robustness and sensitiveness.

4. RESULTS

4.1. RQ1 - Effectiveness

Figure 6 shows box plots that summarize the distributions of the comprehension level by Method, in the four replications (circles are outliers). The distribution of the comprehension level achieved by participants having screen mockups available (Method = S) are highlighted by filled boxes, while the participants having just the textual use cases (Method = T) are represented by hollow boxes. We can observe that the participants achieved a better comprehension level when they accomplished comprehension tasks with the help of screen mockups. Such trend appears consistently in all the four experiments. It is important to notice that though UniBas2 comprehension questionnaires

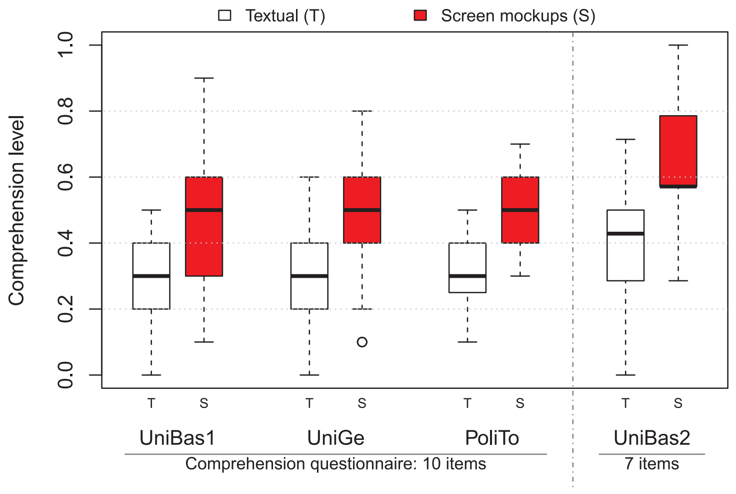


Fig. 6. Box plots for comprehension level (Effectiveness).

Table VII. Descriptive Statistics of Comprehension Level and Analysis Results

		T			S			Wilcoxon	Cliff's
Experiment	N	mean	median	σ	mean	median	σ	p-value	δ
UniBas1	33	0.27	0.30	0.14	0.48	0.50	0.20	<**0.001**	0.60
UniGe	51	0.29	0.30	0.15	0.48	0.50	0.16	<**0.001**	0.63
PoliTo	24	0.32	0.30	0.12	0.49	0.50	0.12	<**0.001**	0.69
UniBas2	31	0.38	0.43	0.16	0.65	0.57	0.17	<**0.001**	0.73

counted just seven items, the results are generally comparable since the comprehension level is normalized by the number of items.

Table VII (leftmost eight columns) reports the descriptive statistics: sample size (N), mean, median, and standard deviation for the comprehension level. We can observe that both mean and median comprehension level for the T group are consistently smaller than those in S group. The variability, expressed as standard deviation, is mostly comparable both across groups and experiments.

4.1.1. Raw Effects. As far as the distribution of the comprehension level variable is concerned, the Shapiro-Wilk test returned p-values smaller than α in all the experiments with the exception of UniBas1 where p-values are 0.32 and 0.056 for S and T, respectively (see Figure 11 in Appendix B).

Table VII (rightmost two columns) reports the p-values for the Wilcoxon test, and the effect size. The results from the Wilcoxon test—all the p-values are consistently smaller than 0.016 (α level after Bonferroni correction)—let us reject the hypothesis H_{c0} for all the experiments. Therefore, the use of screen mockups significantly improves the comprehension of functional requirements. For all the experiments, the effect size is large. The values are in between 0.6 and 0.73.

4.1.2. Knowledge Categories. We analyzed the combined effect of Method with the KD variable that represents the Knowledge Domain. We also analyzed the different categories together with the co-factors—Application and Lab—by means of a permutation test. The results of the permutation test for each individual experiment are reported in the Table XV, in Appendix B. the relationship between S and T and the comprehension

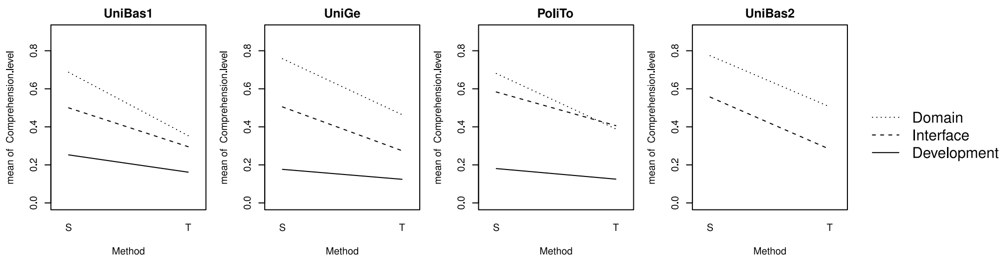


Fig. 7. Interaction plots for Method and Knowledge Domain (KD).

level achieved in the three categories is also represented graphically, by means of interaction plots in Figure 7.

From the permutation tests we can observe that in our experiments Method has a significant effect on the dependent variable (p-values in Table XV are all less than 0.001). This analysis substantially confirms our previous findings: the presence of screen mockups affects the comprehension level. This effect can be also observed in the graphs of Figure 7: all the lines are bent towards the right, the achieved comprehension level is higher when screen mockups are present (S) than when the requirements are purely textual (T). Moreover, this is true for each value of *KD*.

Another result, common to all the experiments, is that the knowledge category (factor *KD* in Table XV of Appendix B) has a statistically significant effect on the comprehension level: the participants, independently from the presence of mockups, achieved a different comprehension level in distinct knowledge domains. This effect can be also observed in Figure 7. In particular, we can observe that the domain line (dotted) is higher than the interface line (dashed), which is higher than the development line (continuous), the latter when present (i.e., in the first three experiments).

We also found a significant interaction between Method and KD (see Table XV) in the first three experiments and not in the fourth. This means that Method may have a different effect for different knowledge domains. This interaction implies that some slopes of the segments in Figure 7 are different. For example, in the UniBas1 graph, the slope of the Domain line is steeper than the Interface line. This implies that the comprehension level improvement caused by screen mockups is larger for Domain with respect to Interface.

4.1.3. Co-factors. The model used for the permutation test also includes the co-factors Application and Lab. In general, no co-factor exercised any direct effect on the dependent variable comprehension level. On the other hand, we observe a significant interaction between Lab and Method in UniGe (see also Figure 12 in Appendix B). That is, the combined effect of Lab and absence/presence of screen mockups in the requirements specification documents is nonadditive on comprehension level. In the PoliTo experiment, we detected a significant interaction between Application and KD (see Figure 13 in Appendix B). This result indicates that on the two experimental objects the participants obtained different comprehension levels for domain, interface, and development. Future work will be devoted to understanding how the comprehension level on these kinds of knowledge is effected by the participants' experience and the kind of software.

4.2. RQ2 - Effort

Figure 8 shows the box plots of task completion time grouped by Method and Experiment. The box plots show that the participants in all the experiments spent slightly less time to accomplish the comprehension task when using screen mockups (see the

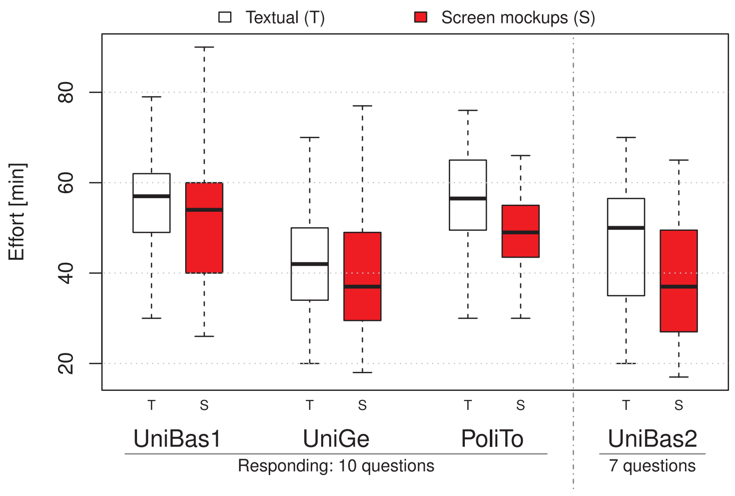


Fig. 8. Box plots for Task completion time (Effort).

Table VIII. Descriptive Statistics of Task Completion Time and Analysis Results

		T			S			Wilcoxon	Cliff's
Experiment	N	mean	median	st. dev.	mean	median	st. dev.	p-value	δ
UniBas1	33	55.00	57	11.76	51.12	54	14.48	0.09	0.18
UniGe	51	42.55	42	12.34	40.00	37	13.68	0.18	0.14
PoliTo	24	55.75	56.5	11.80	49.62	49	9.00	0.06	0.33
UniBas2	31	46.35	50	13.65	38.32	37	13.26	0.03	0.33

filled box plots). It is important to emphasize that the effort measured in experiment UniBas2 cannot be directly compared to that in the other experiments. This is due to the different number of items—seven vs. ten— in the used comprehension questionnaires. In addition, the questionnaires used in UniBas2 were only covered aspects of the problem domain and interactions between the actors and the system, while in the other experiments the participants were asked to answer also questions concerning the solution domain of the subject system.

Table VIII reports descriptive statistics for the time to complete the comprehension tasks. Both mean and median times for the T group are larger than the S group, with differences ranging from 2 to 13 minutes. Standard deviations are all similar and comparable.

4.2.1. Raw Effects. As far as the Effort construct is concerned, we analyzed the Task completion time. The data seem to be non-normally distributed only in one case of UniGe—Shapiro-Wilk test p-value = 0.027 when Method = S (see Figure 11 in Appendix B). To be conservative as much as possible and for uniformity, we opted for nonparametric analyses also on the variable Task completion time. However, we also repeated the analysis for UniBas1, PoliTo, and UniBas2 using parametric tests (i.e., the paired t-test) obtaining consistent results.

Table VIII reports the results of the Wilcoxon test. None of the p-values is smaller than the α level (set to 0.016 for the Bonferroni correction), therefore we cannot reject the null hypothesis H_{t0} in all the experiments in the family. So we can assert that the use of screen mockups does not significantly impact the time to accomplish the comprehension tasks. The effect size is negligible for UniGe, small for UniBas1, and medium for UniBas2 and PoliTo.

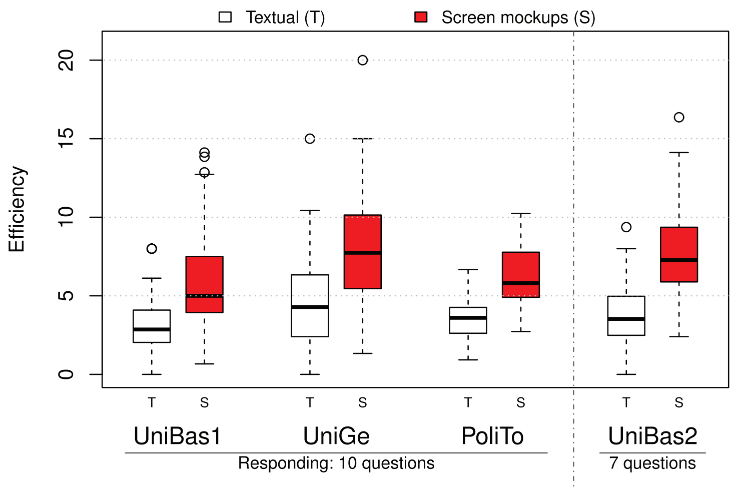


Fig. 9. Box plots for Task efficiency.

Table IX. Descriptive Statistics of Task Efficiency and Analysis Results

		T			S			Wilcoxon	Cliff's
Experiment	N	mean	median	σ	mean	median	σ	p-value	δ
UniBas1	33	3.16	2.86	1.96	6.22	5.00	3.45	<**0.001**	0.59
UniGe	51	4.57	4.29	3.01	7.97	7.74	3.56	<**0.001**	0.56
PoliTo	24	3.57	3.60	1.59	6.22	5.81	2.11	<**0.001**	0.67
UniBas2	31	3.78	3.53	2.03	7.94	7.27	3.24	<**0.001**	0.75

4.2.2. Co-factors. The detailed results of the permutation test are reported in Table XVII in Appendix B. With the only exception of UniGe, we observe that Method has a statistical significant effect. This is in contrast with the result we obtained with the Wilcoxon test reported previously. Such a difference can be explained by considering that the permutation test (as well as ANOVA) report the significance of Method *after* discounting for the (confounding) effect of the other factors (i.e., Lab and Application).

The most important effect that emerges from the permutation test, consistently in all the four experiments, is that on Application. The time required to complete the comprehension task with the AMICO application is larger than with EasyCoin, in all experiments (see also Figure 15 in Appendix B). Moreover, we observe a significant effect of the *Lab* factor in the UniBas1 and UniBas2 experiments.

4.3. RQ3 - Efficiency

Figure 9 shows the box plots of Task efficiency grouped by Method and Experiment. The box plots show that the participants in all the experiments more efficiently accomplished the tasks when using screen mockups (see the filled boxes). Though UniBas2 comprehension questionnaires counted just seven items, the results are generally comparable since the Task efficiency is normalized by the number of items.

Table IX reports the descriptive statistics for Task efficiency. Both mean and median efficiency for group T are roughly half of group S.

4.3.1. Raw Effects. The results of the Shapiro-Wilk test suggest that the data could not be normally distributed in two cases: UniBas1 and UniGe (see Figure 11 in Appendix B). For this reason, we analyzed the Task efficiency by means of nonparametric tests as done for the other two dependent variables. However, we also repeated the analysis for

Table X. Importance of Screen Mockups as Source of Information

		Experiment				
System	Category	UniBas1	UniGe	PoliTo	UniBas2	All
AMICO	Domain	●	●	●	●	●
	IO	●	◐	●	◐	●
	Development	○	○	◐	-	○
	All	●	◐	●	●	●
EasyCoin	Domain	◐	◐	◐	◐	◐
	IO	◐	◐	●	◐	●
	Development	◐	◐	◐	-	◐
	All	◐	◐	●	◐	◐
Both	All	◐	◐	●	●	●

● : predominant, ◐ : relevant, ○ : not relevant.

PoliTo and UniBas2 using parametric tests (i.e., the paired t-test) obtaining consistent results.

Table IX reports the results of the Wilcoxon test. All the p-values are smaller than the α level (0.016 on the basis of the Bonferroni correction) and then we can reject the null hypothesis H_{e0} in all the experiments, that is, the use of screen mockups significantly impacts on efficiency. The effect size is large in all experiments.

4.3.2. Co-factors. We observe that Method has a statistical significant effect in all the experiments (see the results of the permutation test on Application and Lab in Table XIX of Appendix B). This result is in agreement with the results of the Wilcoxon test reported previously.

The result of the permutation test also indicated a significant effect of Application in all the experiments with the only exception of UniGe. We observe that students more efficiently performed tasks on EasyCoin. We also observe a significant effect of Application for UniBas1, PoliTo, and UniBas2.

4.4. Sources of Information

Table X summarizes the relevance of screen mockups as source of information for comprehension tasks. Results are divided by Experiment, Application, and Knowledge Domain. This table reports the cases where screen mockups were predominant source of information as black filled circles, those where mockups were simply relevant as half-filled circles, and those where they were irrelevant as hollow circles. Overall (see the last row of Table X), screen mockups were relevant in UniBas1 and UniGe and predominant in PoliTo and UniBas2.

When analyzing AMICO and EasyCoin separately, we observe two distinct behaviors: screen mockups proved to be predominant more often for AMICO than for EasyCoin. As far as AMICO is concerned, mockups represent consistently across the experiments a fundamental source for domain-related questions, while they are almost not relevant for questions belonging to the development category. Concerning EasyCoin, screen mockups appear relevant, but not predominant in many cases.

4.5. Postexperiment Questionnaire Results

The answers to the postexperiment questionnaire are graphically summarized in Figure 10. Concerning PQ1 and PQ2, we observe the participants agreed on the fact that enough time was allowed to perform comprehension tasks and that the questions

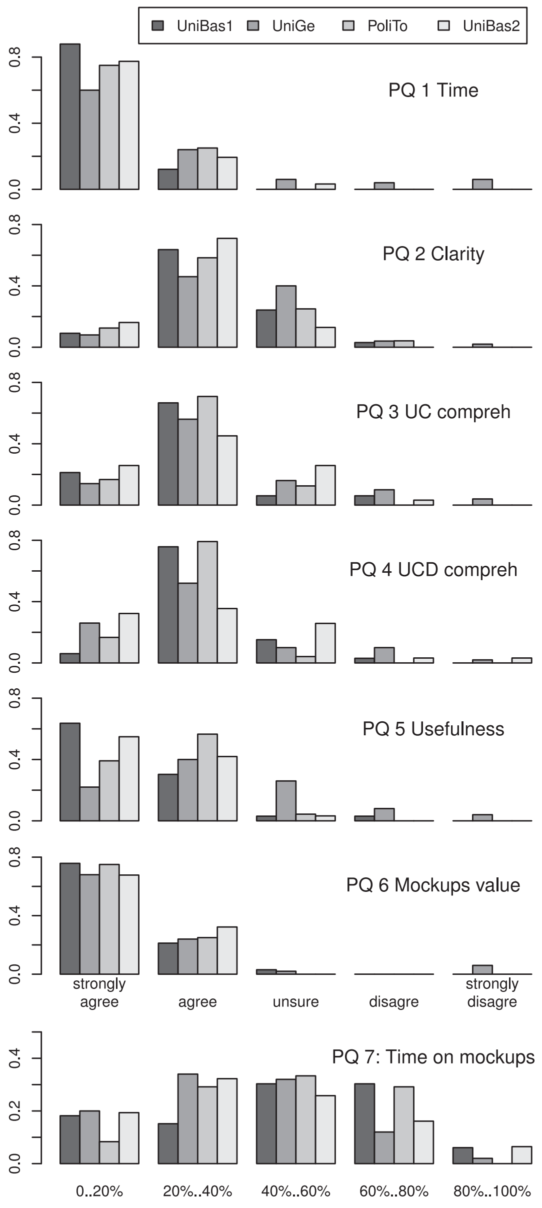


Fig. 10. Postquestionnaire answers.

of the comprehension questionnaire were clearly specified, respectively. The median value of the answers corresponded to “strongly agree” for PQ1 and “agree” for PQ2. The participants were able to understand use cases (PQ3) and use case diagrams (PQ4). For all the experiments, the median values for PQ3 and PQ4 corresponded to “agree”. The participants also acknowledged that the tasks were useful in the context of the course they have been attending (PQ5). The median values corresponded to either “strongly agree” or “agree” in all the experiments.

Table XI. Analysis of Efficiency Variance of All Experiments (Permutations Test)

Error	Factor	Df	R Sum Sq	R Mean Sq	Iter	Pr(Prob)
Subject	Exp	3	127.846	42.615	5000	<**0.001**
	Residuals	135	982.277	7.276		
Subject:Method	Method	1	785.546	785.546	5000	<**0.001**
	Exp:Method	3	17.321	5.774	87	0.989
	Residuals	135	1204.984	8.926		

The main result from the postexperiment questionnaire is that the participants in all the experiments agreed (even strongly about 70% of them) on the usefulness of the screen mockups (PQ6). This result is relevant for the goals of our study.

Moreover, the analysis of the responses to question PQ7 showed that 80% of the participants indicated that they spent more than 20% on screen mockups. The estimated average, computed considering the mid-value of the intervals (e.g., 30% for the interval 20%–40%), in the four experiments was 48%, 38%, 47%, and 42%, respectively.

Summarizing, the analysis of the responses to the postexperiment survey questionnaire suggests the following main outcomes: (i) the experiment material was considered clear and its overall quality good; (ii) screen mockups were judged useful to comprehend functional requirements, and (iii) a considerable amount of time was spent on screen mockups.

4.6. Cross-Experiment Analysis

To have a more complete picture, we also conducted a comprehensive analysis considering all the data from the four experiments together. We conducted a factorial analysis of variance, by means of nonparametric permutation tests, considering as output (Y) the two dependent variables Effectiveness and Efficiency and as factors Method, Experiment (*Exp*) and their first-order interaction. We excluded the Effort variable from the analysis since the comprehension time is not comparable among the experiments: in UniBas2, reduced comprehension questionnaires were used.

The linear model used for the analysis is the following:

$$Y = \beta_0 + \beta_1 \cdot \mathit{Exp} + \beta_2 \cdot \mathit{Method} + \beta_3 \mathit{Exp} \cdot \mathit{Method} + \epsilon(\mathit{Subject}) + \epsilon,$$

where Y can be either Effectiveness or Efficiency and β_i represent a set of linear coefficients. In addition, since this is a repeated measure model, we have an error term, $\epsilon(\mathit{Subject})$, that depends on the participant—measured two times associated with distinct values of Method—and the usual error term.

Table XI shows the permutation test results for $Y =$ Task Efficiency. We can observe a statistically significant effect on Method and Experiment, but no significant interaction between these two factors. The analysis for task efficiency substantially confirms the findings on individual experiments reported previously.

To complete the picture, we also conducted permutation tests for the other dependent variable, Effectiveness—measured as Comprehension level. The tests revealed a significant effect of mockups (see Table XX in Appendix B), no significant differences among the experiments, and no interaction between the two factors.

5. DISCUSSION

5.1. Summary of Results

Running a family of experiments rather than an individual experiment provided us with more evidence about external validity—including generalizability—of the results [Basili et al. 1999]. The same hypotheses were tested and confirmed in three distinct contexts employing four distinct profiles of participants (see Table XII). Each replication provided further evidence supporting the confirmation of the hypotheses.

The results observed in each experiment are consistent with each other: (i) comprehension of functional requirements improves when use cases are augmented with screen mockups; (ii) the presence of screen mockups in requirements specification documents does not affect task completion time; and (iii) a positive increment of efficiency was observed when using screen mockups. The main results of our family of experiments are summarized in Table XII. The study also opens a number of interesting discussion points.

—*Comprehension Effectiveness*. We observed a significantly large effect of the presence of screen mockups on the comprehension level. The mean percentage improvement of comprehension level achieved with mockups ranges from 56% (for PoliTo) to 78% (for UniBas1), with an average of 69%.

—*Knowledge Category*. We observed that the comprehension effectiveness improvement varied on the specific Knowledge category addressed. In particular, the largest benefits were achieved for problem domain and interface. Development category obtained the smallest benefits. We mostly expected a small or no influence on development knowledge, since the object of the investigation is requirement specification technique that, by definition, focuses on "what" the system under consideration should do and not on "how" it should be implemented [Bruegge and Dutoit 2003]. Conversely, we had no previous expectations concerning the domain aspects: from these results, we learn that screen mockups are actually useful in shedding light on the problem domain in addition to the interface of the system.

—*Effort*. The magnitude of the reduction in time due to screen mockups usage is medium in two of the four conducted experiments (PoliTo and UniBas2). While in the other two cases, the effect size is small (UniBas1) or negligible (UniGe). The mean time reduction varies between 6% (for UniGe) and 17% (for UniBas2). We observed that the Application co-factor has an effect on task completion time larger than that of Method. In other words, the—nonlarge—effect of mockups has been confounded by the "noise" introduced by the difference among the two objects used in the experiment—the applications AMICO and EasyCoin.

—*Efficiency*. We observed a significant large effect of screen mockups on the efficiency with which the participants accomplished comprehension tasks. The average improvement ranged from 74% (for PoliTo) up to 110% (for UniBas2). That is, efficiency almost doubles when screen mockups are used together with use cases.

—*Participants' Profile*. Each participant category positively benefited in terms of comprehension and efficiency when using screen mockups together with use cases. Hence, benefits stemming from the presence of mockups seem to be independent from the participants' profile. However, a slightly larger benefit—in term of both comprehension and efficiency—has been observed for UniBas2, while a slightly smaller one for PoliTo. This may indicate that screen mockups are more useful for the participants with lowest experience in modeling.

—*Lab*. The effect of Lab observed in UniBas1 and UniBas2 on Task completion time could indicate a possible learning effect in these experiments (the participants used longer time in the first run than in the second one). As for the other two dependent variables, the effect of Lab is statistically significant neither in UniBas1 and UniBas2 nor in UniGe and PoliTo. However, in UniGe, a significant interaction between Lab and Method was shown with respect to comprehension level.

—*Application*. In all the experiments in our family, we observed a statistically significant effect of Application on Task completion time. Although the requirement specification documents for AMICO and EasyCoin were similar in size and complexity, the

Table XII. Summary of the Results

Experiment	Description/ differences	# Participants	Participant's profile	H_{ca} accepted? (effectiveness)	H_{ta} accepted? (effort)	H_{ea} accepted? (efficiency)
UniBas1	Original Experiment	33	2nd CS Undergraduate	yes	no	yes
UniGe	Replication of UniBas1 - Different participants - Random assignment - No break	51	3rd CS Undergraduate	yes	no	yes
PoliTo	Replication of UniBas1 - Different participants	24	2nd CE Graduate	yes	no	yes
UniBas2	Replication of UniBas1 - Different participants - Limited questionnaire	31	1st M+T Graduate	yes	no	yes

(CS = Computer Science, CE = Computer Engineering, M = Maths, T = Telecommunications).

time required to complete the comprehension task with AMICO was larger (see also Figure 15 in Appendix B). We attribute this result to the differences in the domain of the two applications. Apparently, the participants were more familiar with EasyCoin. This point was somewhat confirmed because of the significant effect of Application on Task efficiency. This result held for all the experiments in our family with only the exception of UniGe. As a lesson learned, we remark the fact that the sheer objective measures of size and/or complexity alone are not enough to characterize the difficulty or workload of a task, they should be complemented with subjective familiarity with task and materials.

—*Source of Information*. Screen mockups represent a relevant and sometimes predominant source of information for the comprehension of functional requirements. We notice a difference between AMICO and EasyCoin. The participants found screen mockups more useful to accomplish the comprehension task on AMICO. That difference could be attributed to the domain difference among the two applications. Mockups have been less consulted and deemed less useful for the application—EasyCoin—which the participants were more familiar with.

—*Postexperiment Questionnaire*. The perception of participants confirmed the objective findings of the experiments: screen mockups are useful to improve the comprehension of functional requirements. Furthermore, the participants found that, to accomplish a comprehension task, they spent on average about 40% of the total time to analyze mockups. This result, together with the mean improvement in comprehension level (about 69%), the average reduction of the time to accomplish the tasks (about 10%), and the mean improvement in efficiency (about 89%), supports the conjecture that use cases augmented with screen mockups convey knowledge in a more effective and efficient way.

5.2. Implications

We adopted a perspective-based approach to judge the practical implications of our study, taking into consideration the *practitioner/consultant* (simply "practitioner", from here on) and *researcher* perspectives suggested by Kitchenham et al. [2008].

—Augmenting use cases with computer-drawn screen mockups yields an average improvement in term of comprehension of circa 69%. Such effect of screen mockups is statistically significant. This result is relevant for the practitioner. The question of whether (or not) to use screen mockups for complementing use cases has so far not been investigated through controlled experiments. Thus, the contribution of our family of experiments is a first indication on the usefulness of screen mockups to improve the comprehension of functional requirements. This result is relevant from the researcher perspective since our study poses the basis for future work.

—The presence of screen mockups induces no additional time burden, that is, mockups are not a cause of distraction for stakeholders. Indeed, screen mockups slightly reduce the comprehension time for each kind of participants. Also this result is relevant for the practitioner and the researcher.

—The presence of screen mockups is able to almost double the efficiency of comprehension tasks of the functional requirements. This result is particularly relevant for the practitioner interested to use screen mockups in conjunction with use cases in their companies.

—The effect of computer-drawn screen mockups on the comprehension of functional requirements (represented with use cases) has been empirically assessed for the first time in our investigation. This is relevant for the researcher, who could be interested in assessing why mockups improve the comprehension of functional requirements. Our family of experiments poses the basis for this kind of investigations.

—Screen mockups proved to be a relevant source of information for comprehending features concerning the application domain and the interaction between users and system. Mockups constitute a relevant or predominant source of information to understand the problem domain of a subject system. Screen mockups represent a predominant source of information, when stakeholders are not familiar with the domain. This finding is relevant from both the practitioner and the researcher perspectives. The practitioner could decide to use screen mockups in all those projects in which the involved stakeholders are not particularly familiar with the application domain of the software to be developed. The researcher could be interested in assessing how familiarity affects benefits stemming from the use of the screen mockups.
—The usage of screen mockups could reduce the number of defects originates in the requirements engineering process and could also positively impact communication among stakeholders. This implication is very relevant for the practitioner.
—The study is focussed on desktop applications for cataloguing collections of coins and for the management of condominiums. From the researcher perspective, the effect of mockups on different type of applications (e.g., Web applications) represents a possible future direction.
—The requirements specification documents could be considered as developed in the following kinds of projects: in-house software, product, or contract [Lauesen 2002]. The magnitude of the benefits deriving from the presence of screen mockups suggests that the obtained result could be generalized for other kinds of software projects. This point deserves further investigations and it is relevant for both the practitioner and the researcher.
—We considered here screen mockups developed with Pencil. Using different computer tools or hand-drawn mockups could lead to different results. This aspect is relevant for practitioners, interested in understanding the best way for building mockups, and researchers, interested in investigating why a different method/tool should affect requirements comprehension.
—The benefits deriving from screen mockups appear to be independent from the participants' profile and modeling experience. This result is practically relevant for the industry: stakeholders with different profiles and skills can benefit from the presence of mockups. This aspect is clearly relevant also for the researcher.
—The requirements specification documents were realistic enough for small-sized software projects. Although we are not sure that the achieved results scale to real projects, the magnitude of the benefits stemming from the presence of screen mockups reassures us that the outcomes might be generalized to larger projects.
—The use of computer-drawn screen mockups does not require a complete and radical process change in an interested company. This assertion can be deduced from the results of the study presented here and those presented by Ricca et al. [2010c]. This find can be considered relevant for the practitioner.
—The adoption of screen mockups should also consider the cost for their creation. This aspect, not considered in this article, is of particular interest for the practitioner. In fact, it would be crucial knowing whether the additional cost needed to develop screen mockups is adequately paid back by the improved comprehension of functional requirements. In that direction, we conducted a preliminary empirical investigation and its results are reported in the paper by Ricca et al. [2010c]. Bachelor and Master students from the University of Basilicata and from the University of Genova, and software professionals were involved. We gathered information about the time to define and to design screen mockups with the Pencil tool and then we collected the participants' perceptions about the effort to build screen mockups. The main finding of that study indicated that: the participants perceived screen mockups construction

as a non-onerous activity that requires a low effort as compared with the effort needed to develop use cases. Such preliminary findings should be confirmed with more detailed studies in order to precisely quantify costs and benefits of augmenting requirements with screen mockups. This point is relevant also for the researcher, who could be interested in defining models to predict the effort needed to develop screen mockups [Scanniello et al. 2013].

—The diffusion of a new technology/method is made easier when empirical evaluations are performed and their results show that such a technology/method solves actual issues [Pfleeger and Menezes 2000]. This is why the results of our family of experiments could increase the diffusion of screen mockups in the software industry. This is of particular interest for the practitioner.

6. RELATED WORK

This section discusses the related literature concerning experiments aimed at: (i) assessing understandability of requirements specifications/models and (ii) assessing if and how graphical elements improve the understanding of software artifacts.

6.1. Understandability of Requirements Specifications/Models

The results of a mapping study by Condori-Fernandez et al. [2009]—a key instrument of the evidence-based paradigm—suggest that: (i) understandability is the most commonly evaluated aspect of software requirements specification approaches, (ii) experiments are the most commonly used research method, and (iii) the academic environment is where most empirical evaluation takes place. Condori-Fernandez et al. [2009] base these results on 19 empirical studies concerning understandability of requirements specifications/models. With respect to these results, our work falls fully in the category of works investigating understandability of requirements specifications/models by means of controlled experiments with students.

Anda et al. [2001] perform an exploratory study with 139 undergraduate students divided in groups to investigate three different sets of guidelines to construct and document use case models. Each group used one out of the three sets of guidelines when constructing a use case model from an informal requirements specification. After completing the use case model, each student was asked to answer a questionnaire. The analysis on the data gathered from that questionnaire reveals that guidelines based on templates are easier to understand. Furthermore, guidelines based on templates better support students in constructing use case models and the quality of the produced documentation is higher. As in our experiments, the understanding of the requirements when using different guidelines in the construction of use case models was assessed by means of a questionnaire even if the authors used a different way to measure the correctness of the answers to the questionnaires.

The need of writing guidelines to specify clear and accurate use cases is evident also from the results of the study by Phalp et al. [2007]. In that work, the authors describe an empirical work to assess the utility of two requirements writing guidelines, namely CREWS and CP. In particular, the use cases are assessed using a check-list previously presented by the authors. The results of the study reveal that these techniques are comparable in terms of their ability to produce clear and accurate use cases.

Kamsties et al. [2003] present the results of an experiment on understandability of black-box and white-box requirements models from the point of view of a customer. In a black-box style, a system is described by its externally visible behavior, while in a white-box style, a system is described by the behavior of its constituent entities (e.g., components or objects). The dependent variables used in that study are the same as ours and the comprehension was measured by means of questionnaires as we did. The authors measured the correctness of the answers to the questionnaires as we did, that is, using a correct/wrong approach. The results of the study confirm the general advice that requirements should only describe the externally visible behavior of systems. That is, black-box specifications are better than white-box specifications.

Zimmerman et al. [2002] present the results of an empirical study concerning understandability of requirements specifications/models. The goal is to determine how some factors regarding a formal method (i.e., state-based requirements specification language) affect the requirements readability of aerospace applications. The study is conducted using students with computer science and aerospatial engineering backgrounds. The results of a data analysis reveal that familiarity with state machines is a much more influential factor than the familiarity with the problem domain of the system under study.

Abrahão et al. [2013] present the results of a family of experiments to investigate whether the comprehension of functional requirements is influenced by the use of UML sequence diagrams. The family consists of an experiment [Gravino et al. 2008] and four replications, which are performed in different locations with students and professionals with different abilities and levels of experience with the UML. The main result of the family is that the use of sequence diagrams improves the comprehension of functional requirements only in the case of high ability and more experienced participants: a given experience/ability is needed to get an improved comprehension of functional requirements. This latter result contrasts somewhat with the findings observed in this article: all the kinds of participants got an improved comprehension of functional requirements when the treatment is administered.

6.2. Usefulness of Graphical Elements

Almost all the related papers found in literature (except, e.g., Ricca et al. [2010a]) show that the participants benefit from the use of graphical elements. We intentionally intend the term "graphical element" in a broad sense. Tables, diagrams, icons,[7] ad-hoc symbols and pictures fall in this category. For example, Ricca et al. [2009] empirically investigate through controlled experiments the impact of FIT tables on the clarity of requirements. FIT tables are simple HTML tables serving as the input and expected output for the acceptance tests [Mugridge and Cunningham 2005]. Two experiments with students were executed to assess the impact of FIT tables on the clarity of functional requirements. To assess the possible benefit deriving from the use of FIT tables, the participants were provided with either text-only requirements or textual requirements plus FIT tables. Similarly to the experiments presented in this article, comprehension was assessed by means of questionnaires and the time to answer to the questionnaire was recorded directly by participants. The results indicate that: (i) FIT tables help the comprehension of functional requirements and (ii) the overhead induced by the presence of FIT tables—measured in terms of time—is negligible.

Andrew and Drew [2009] empirically investigate whether use case diagrams improve the effectiveness of use cases by providing visual clues. The investigation is conducted with a group of 49 students, who are provided with only use cases (control group) or with use cases and use case diagrams (treatment group). The results show that students employing both a use case diagram and use cases achieve a significantly higher level of understanding. Similarly to our study, the results suggest the combined use of visual representations and use cases.

Bratthall and Wohlin [2002] experimentally study the usefulness of graphical elements to support understanding and maintenance of evolving software. That study involves 35 people, who have software experience (i.e., master and Ph.D students). The

[7]In the UML a stereotype can be rendered in two ways: as a name enclosed by <<>> or by a special conceived icon.

authors compared 10 simple different representations aiming at enriching the design of a software architecture with graphical elements. The graphical elements added to UML/SDL diagrams are rectangles visually representing component control relationship, internal/external complexity relationship and component size relationship. The results indicate that graphical elements can be useful to control evolution. They can be also viewed as warning signs or useful methods for classifying software components. There are a lot of commonalities between this experiment and ours and also the results go in the same direction: graphical elements help in understanding. Unlike our family of experiments, the time to accomplish a comprehension task is not considered.

Hendrix et al. [2002] present two experiments to investigate the influence of Control Structure Diagram (CSD) on the comprehension of Java source code. CSD is a graphical notation able to represent some constructs of programming languages (i.e., sequence, selection, and iteration) by means of graphical symbols. The experimental design focuses on fine-grained comprehension of source code. The comprehension achieved by participants on the source code was assessed through a questionnaire. However, the time to accomplish a comprehension task was automatically recorded by a Web application. The results show that the use of CSD improves the performance of the participants (better comprehension and reduced task completion time).

Staron et al. [2006] present a series of controlled experiments about UML stereotypes represented by means of ad-hoc icons. The participants in these experiments were both students and professionals. The authors assess the effectiveness of stereotyped classes in UML class diagrams to comprehend object-oriented applications in the telecommunication domain. They show that the use of icons significantly helps both kinds of participants (i.e., students and professionals) to improve comprehension.

Ricca et al. [2010a] conduct a family of experiments with graduate, undergraduate, and Ph.D students on the understanding of UML class diagrams with and without Conallen's stereotypes expressed with icons [Conallen 2002]. Unlike in the study by Staron et al. [2006], icons seem little useful in understanding. The main finding of that experiment is that icons reduce the gap between experienced and less experienced participants. This is the only empirical study among those considered, where graphical elements do not help in improving the comprehension.

There are several differences among the studies discussed in the latter two sections and our family of experiments. The most remarkable difference is the goal. In fact, we focused on the usage of screen mockups to improve the comprehension of functional requirements represented with use cases. All the other studies, with the exception those by Ricca et al. [2009] and Andrew and Drew [2009], concern lower-level artifacts (i.e., design models and code).

7. CONCLUSION AND FUTURE WORK

In this article, we report on the results from a family of four controlled experiments designed to assess whether participants—having different profiles—benefit from the availability of screen mockups in the comprehension of functional requirements represented with use cases. The results indicate that the addition of screen mockups has a large statistically significant effect on both the functional requirements comprehension effectiveness (+69% improvement on average) and the efficiency to accomplish comprehension tasks (+89% improvement on average). The effect on the time to accomplish a comprehension task is not statistically significant. Such results were consistently observed for all of the four experiments in the family.

Possible future directions for our research are: (i) validating the results within industrial settings by executing case studies; (ii) evaluating the effect of providing information to participants in an incremental way; and (iii) assessing whether the additional effort and cost required to develop screen mockups are adequately paid back by the improved comprehension of functional requirements, for example, by pursuing the studies presented by Ricca et al. [2010c] and Scanniello et al. [2013].

ELECTRONIC APPENDIXES

The electronic appendixes for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENTS

The authors would like to thank the participants in our experiments.

REFERENCES

S. Abrahão, C. Gravino, E. Insfran, G. Scanniello, and G. Tortora. 2013. Assessing the effectiveness of sequence diagrams in the comprehension of functional requirements: Results from a family of five experiments. *IEEE Trans. Softw. Eng.* 29, 3, 327–342.

S. Adolph, A. Cockburn, and P. Bramble. 2002. *Patterns for Effective Use Cases*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

A. Agresti. 2007. *An Introduction to Categorical Data Analysis*. Wiley-Interscience.

B. Anda, D. I. K. Sjøberg, and M. Jørgensen. 2001. Quality and understandability of use case models. In *Proceedings of the European Conference on Object-Oriented Programming.* Berlin, Springer-Verlag, 402–428.

G. Andrew and P. Drew. 2009. Use case diagrams in support of use case modeling: Deriving understanding from the picture. *J. Datab. Manage.* 20, 1, 1–24.

J. Aranda, N. Ernst, J. Horkoff, and S. Easterbrook. 2007. A framework for empirical evaluation of model comprehensibility. In *Proceedings of the ICSE Workshop on Modeling in Software Engineering*. IEEE, 7–13.

E. Astesiano, M. Cerioli, G. Reggio, and F. Ricca. 2007. Phased highly-interactive approach to teaching UML-based software development. In *Proceedings of the Symposium at MODELS 2007*. 9–19.

E. Astesiano and G. Reggio. 2012. A disciplined style for use case based requirement specification. Tech. Rep. DISI-TR-12-04. DISI University of Genova, Italy. http://softeng.disi.unige.it/TR/Disciplined Requirements.pdf.

V. Basili, G. Caldiera, and D. H. Rombach. 1994. *The Goal Question Metric Paradigm, Encyclopedia of Software Engineering*. Wiley.

V. Basili, F. Shull, and F. Lanubile. 1999. Building knowledge through families of experiments. *IEEE Trans. Softw. Eng.* IEEE, 456–473.

L. Bratthall and C. Wohlin. 2002. Is it possible to decorate graphical software design and architecture models with qualitative information?-An Experiment. *IEEE Trans. Softw. Eng.* 28, 12, 1181–1193.

B. Bruegge and A. H. Dutoit. 2003. *Object-Oriented Software Engineering: Using UML, Patterns and Java,* 2nd Ed. Prentice-Hall.

J. Carver, L. Jaccheri, S. Morasca, and F. Shull. 2003. Issues in using students in empirical studies in software engineering education. In *Proceedings of the International Symposium on Software Metrics*. IEEE Computer Society, Los Alamitos, CA, 239–250.

M. Ceccato, P. Di Penta, P. Falcarin, F. Ricca, M. Torchiano, and P. Tonella. 2014. A family of experiments to assess the effectiveness and efficiency of source code obfuscation techniques. *Empir. Softw. Eng.* 19, 4, 1040–1074.

M. Ciolkowski, D. Muthig, and J. Rech. 2004. Using academic courses for empirical validation of software development processes. In *Proceedings of the EUROMICRO Conference*. IEEE Computer Society, Los Alamitos, CA, 354–361.

N. Cliff. 1993. Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychol. Bull.* 114, 3, 494–509.

A. Cockburn. 2000. *Writing Effective Use Cases*. Addison-Wesley, Reading, MA.

J. Conallen. 2002. *Building Web Applications with UML*, Second Edition. Addison-Wesley, Reading, MA.

N. Condori-Fernandez, M. Daneva, K. Sikkel, R. Wieringa, O. Dieste, and O. Pastor. 2009. Research findings on empirical evaluation of requirements specifications approaches. In *Proceedings of the Workshop on Requirements Engineering*. Valparaiso University Press, 121–128.

O. J. Dunn. 1961. Multiple comparisons among means. *J. ASA*, 56, 52–64.

J. Ferreira, J. Noble, and R. Biddle. 2007. Agile Development Iterations and UI Design. In *Proceedings of the Agile Conference (AGILE)*. 50–58.

M. Fowler. 2003. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3rd Ed. Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

C. Gravino, G. Scanniello, and G. Tortora. 2008. An empirical investigation on dynamic modeling in requirements engineering. In *Proceedings of the Conference on Model Driven Engineering Languages and Systems*. IEEE Computer Society, Los Alamitos, CA, 615–629.

J. Hannay and M. Jørgensen. 2008. The role of deliberate artificial design elements in software engineering experiments. *IEEE Trans. Softw. Eng.* 34, 2, 242–259.

H. R. Hartson and E. C. Smith. 1991. Rapid prototyping in human-computer interface development. *Interacting with Computers* 3, 1, 51–91.

D. Hendrix, J. Cross, and S. Maghsoodloo. 2002. The effectiveness of control structure diagrams in source code comprehension activities. *IEEE Trans. Softw. Eng.* 28, 5, 463–477.

J. J. Higgins. 2004. *An Introduction to Modern Nonparametric Statistics*. Brooks/Cole.

L. Hochstein, V. R. Basili, M. V. Zelkowitz, J. K. Hollingsworth, and J. Carver. 2005. Combining self-reported and automatic data to improve programming effort measurement. *ACM SIGSOFT Softw. Eng. Notes* 30, 5, 356–365.

K. Hogarty and J. Kromrey. 2001. We've been reporting some effect sizes: Can you guess what they mean? In *Proceedings of the Annual Meeting of the American Educational Research Association*.

L. Huang and M. Holcombe. 2009. Empirical investigation towards the effectiveness of test first programming. *Inf. Softw. Tech.* 51, 1, 182–194.

W. S. Humphrey. 1995. *A Discipline for Software Engineering*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

ISO. 1991. Information technology–software product evaluation: Quality characteristics and guidelines for their use, *ISO/IEC IS 9126*, ISO, Geneva.

ISO. 2000. Ergonomic requirements for office work with visual display terminals (VDTs) – Part 9: Requirements for non-keyboard input devices. *ISO 9241-11*, ISO, Geneva, Switzerland.

ISO. 2011. Systems and software engineering–Systems and software quality requirements and Evaluation (SQuaRE)–System and software quality models. *ISO/IEC 25010*, ISO, Geneva, Switzerland.

I. Jacobson. 2004. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley Longman Publishing Co., Inc., Redwood City, CA.

N. Juristo and A. M. Moreno. 2001. *Basics of Software Engineering Experimentation*. Kluwer Academic Publishers, Englewood Cliffs, NJ.

R. I. Kabacoff. 2011. *R in Action – Data Analysis and Graphics with R*. Manning Publications Co., Shelter Island, NY.

E. Kamsties, A. von Knethen, and R. Reussner. 2003. A controlled experiment to evaluate how styles affect the understandability of requirements specifications. *Inf. Softw. Tech.* 45, 14, 955–965.

B. Kitchenham, H. Al-Khilidar, M. Babar, M. Berry, K. Cox, J. Keung, F. Kurniawati, M. Staples, H. Zhang, and L. Zhu. 2008. Evaluating guidelines for reporting empirical software engineering studies. *Empir. Softw. Eng.* 13, 97–121.

B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. El Emam, and J. Rosenberg. 2002. Preliminary guidelines for empirical research in software engineering. *IEEE Trans. Softw. Eng.* 28, 8, 721–734.

J. A. Landay and B. A. Myers. 1995. Interactive sketching for the early stages of user interface design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press/Addison-Wesley Publishing Co., New York, NY, 43–50.

S. Lauesen. 2002. *Software Requirements: Styles and Techniques*. Addison-Wesley.

M. Lindvall, I. Rus, F. Shull, M. V. Zelkowitz, P. Donzelli, A. M. Memon, V. R. Basili, P. Costa, R. T. Tvedt, L. Hochstein, S. Asgari, C. Ackermann, and D. Pech. 2005. An evolutionary testbed for software technology evaluation. *Innov. Syst. Softw. Eng.* 1, 1, 3–11.

M. G. Mendonça, J. C. Maldonado, M. C. F. de Oliviera et al. 2008. A framework for software engineering experimental replications. In *Proceedings of the 13th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'08)*. IEEE Computer Society, Los Alamitos, CA, 203–212.

B. Meyer. 1985. On formalism in specification. *IEEE Softw.* 3, 1, 6–25.

H. Motulsky. 2010. *Intuitive Biostatistics: A Non-Mathematical Guide to Statistical Thinking*. Oxford University Press.

R. Mugridge and W. Cunningham. 2005. *Fit for Developing Software: Framework for Integrated Tests*. Prentice-Hall.

M. O'Docherty. 2005. *Object-Oriented Analysis and Design: Understanding System Development with UML 2.0* 1st Ed. Wiley.

A. N. Oppenheim. 1992. *Questionnaire Design, Interviewing and Attitude Measurement*. Pinter, London, UK.

S. L. Pfleeger and W. Menezes. 2000. Marketing technology to software practitioners. *IEEE Softw.* 17, 1, 27–33.

K. T. Phalp, J. Vincent, and K. Cox. 2007. Improving the quality of use case descriptions: Empirical assessment of writing guidelines. *Softw. Qual. Cont.* 15, 4, 383–399.

R Core Team. 2013. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

F. Ricca, M. Di Penta, M. Torchiano, P. Tonella, and M. Ceccato. 2010a. How developers' experience and ability influence web application comprehension tasks supported by UML stereotypes: A Series of Four Experiments. *IEEE Trans. Softw. Eng.* 36, 96–118.

F. Ricca, G. Scanniello, M. Torchiano, G. Reggio, and E. Astesiano. 2010b. On the effectiveness of screen mockups in requirements engineering: results from an internal replication. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*. ACM, New York, NY.

F. Ricca, G. Scanniello, M. Torchiano, G. Reggio, and E. Astesiano. 2010c. On the effort of augmenting use cases with screen mockups: results from a preliminary empirical study. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*. ACM, New York.

F. Ricca, M. Torchiano, M. Di Penta, M. Ceccato, and P. Tonella. 2009. Using acceptance tests as a support for clarifying requirements: A series of experiments. *Inf. Softw. Tech.* 51, 2, 270–283.

J. Romano, J. D. Kromrey, J. Coraggio, and J. Skowronek. 2006. Appropriate statistics for ordinal level data: Should we really be using t-test and Cohen's d for evaluating group differences on the NSSE and other surveys? In *Proceedings of the Annual Meeting of the Florida Association of Institutional Research*.

G. Scanniello, C. Gravino, M. Genero, J. A. Cruz-Lemus, and G. Tortora. 2014. On the impact of UML analysis models on source code comprehensibility and modifiability. *Trans. Softw. Eng. Meth.* 23, 2, 13:1–13:26.

G. Scanniello, C. Gravino, and G. Tortora. 2010. Investigating the role of UML in the software modeling and maintenance - A preliminary industrial survey. In *Proceedings of ICEIS*. 141–148.

G. Scanniello, F. Ricca, M. Torchiano, C. Gravino, and G. Reggio. 2013. Estimating the effort to develop screen mockups. In *Proceedings of the EUROMICRO Conference on Software Engineering and Advanced Applications*. 341–348.

S. Shapiro and M. Wilk. 1965. An analysis of variance test for normality. *Biometrika* 52, 3–4, 591–611.

F. Shull, J. C. Carver, S. Vegas, and N. J. Juzgado. 2008. The role of replications in empirical software engineering. *Empir. Softw. Eng.* 13, 2, 211–218.

M. Staron, L. Kuzniarz, and C. Wohlin. 2006. Empirical assessment of using stereotypes to improve comprehension of UML models: A set of experiments. *J. Syst. Softw.* 79, 5, 727–742.

M. Torchiano. 2014. Effsize: Efficientc Effects Size Computation. R package version 1.2-2.

M. Torchiano, F. C. A. Tomassetti, F. Ricca, A. Tiso, and G. Reggio. 2013. Relevance, benefits, and problems of software modelling and model driven techniques–A survey in the Italian industry. *J. Syst. Softw.* 86, 8, 2110–2126.

H. van der Linden, G. Boers, H. Tange, J. Talmon, and A. Hasman. 2003. Proper: A multi disciplinary EPR system. *Int. J. Med. Inf.* 70, 2–3, 149–160.

B. Wheeler. 2010. lmPerm: Permutation Tests for Linear Models. R package version 1.1-2.

C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. 2012. *Experimentation in Software Engineering*. Springer.

R. Young. 2001. *Effective Requirements Practice*. Addison-Wesley, Boston, MA.

T. Yue, L. C. Briand, and Y. Labiche. 2009. A use case modeling approach to facilitate the transition towards analysis models: Concepts and empirical evaluation. In *Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems (MODELS'09)*. Springer-Verlag, Berlin, 484–498.

M. K. Zimmerman, K. Lundqvist, and N. Leveson. 2002. Investigating the readability of state-based formal requirements specification languages. In *Proceedings of the International Conference on Software Engineering*. ACM, New York, 33–43.

Received June 2013; revised September 2013; accepted March 2014