Engineering Requirements with *Desiree*: An Empirical Evaluation

Feng-Lin Li[1], Jennifer Horkoff[2], Lin Liu[3],
Alex Borgida[4], Giancarlo Guizzardi[5], and John Mylopoulos[1]

[1]: University of Trento, Trento, Italy
[2]: City University, London, UK
[3]: Tsinghua University, Beijing, China
[4]: Rutgers University, New Brunswick, USA
[5]: Federal University of Espirito Santo, Vitoria, Brazil

Abstract. The requirements elicited from stakeholders suffer from various afflictions, including informality, vagueness, incompleteness, ambiguity, inconsistencies, and more. It is the task of the requirements engineering process to derive from these a formal specification that truly captures stakeholder needs. The *Desiree* requirements engineering framework supports a rich collection of refinement operators through which an engineer can iteratively transform stakeholder requirements into a specification. The framework includes an ontology, a formal representation for requirements, as well as a tool and a systematic process for conducting requirements engineering. This paper reports the results of a series of empirical studies intended to evaluate the effectiveness of *Desiree*. The studies consist of three controlled experiments, where students were invited to conduct requirements analysis using textbook techniques or our framework. The results of the experiments offer strong evidence that with sufficient training, our framework indeed helps users conduct more effective requirements analysis.

Keywords: Requirements Problem, Controlled Experiment, Hypothesis Testing, Effect Size

1 Introduction

Upon elicitation, requirements are typically mere informal approximations of stakeholder needs that the system-to-be must fulfill. The core Requirements Engineering (RE) problem is to transform these requirements into a specification that describes formally and precisely the functions and qualities of the system-to-be. This problem has been elegantly characterized by Jackson and Zave [14] as finding the specification S that for certain domain assumptions DA entails given requirements R, and was formulated as $DA, S \models R$. Here DA circumscribes the domain where S constitutes a solution for R.

The RE problem is compounded by the very nature of the requirements elicited from stakeholders. There is much evidence that stakeholder requirements (written in natural language) are often ambiguous, incomplete, unverifiable, conflicting, or just plain wrong [11][15]. More specifically, in our earlier studies on the PROMISE requirements dataset [22], we found that 3.84% of the 625 (functional and non-functional) requirements are ambiguous [18], 25.22% of the 370 non-functional requirements (NFRs) are vague, and 15.17% of the NFRs are potentially unattainable (e.g., they implicitly or explicitly use universals like "*any*" as in "any time") [19].

Our *Desiree* framework tackles the RE problem in its full breadth and depth. In particular, it addresses issues of ambiguity (e.g., "notify users with email", where "*e-mail*" may be a means or an attribute of user), incompleteness (e.g., "sort customers", in ascending or descending order?), unattainability (e.g., "the system shall remain operational at *all* times") and conflict (e.g., "high comfort" vs. "low cost"). The *Desiree* framework includes a modelling language for representing requirements (e.g., *DA*, *S*, and *R*), as well as a set of requirement operators that support the incremental transformation of requirements into a formal, consistent specification. The refinement and operationalization operators strengthen or weaken requirements to transform what stakeholders say they want into a realizable specification of functions and qualities.

Requirement operators provide an elegant way for going from informal to formal, from inconsistent/unattainable to consistent, also from complex to simple. To support incremental refinement, we have proposed a description-based representation for requirements [18][19]. Descriptions, inspired by AI frames and Description Logics (DL) [5], have the general form "*Concept* $<slot_1$: $D_1>$ ... $<$ $slot_n$: $D_n>$", where D_i restricts $slot_i$; e.g., R_1 := "Backup <actor: {the_system}> <object: Data> <when: Weekday>". This form offers intuitive ways to strengthen or weaken requirements. For instance, R_1 can be strengthened into "Backup ... <object: Data> <when: {Mon, Wed, Fri}>", or weakened into "Backup ... <object: Data> <when: Weekday $\vee$ {Sat}>". Slot-description (*SlotD*) pairs "$<$ $slot$: $D>$" allow nesting, hence "<object: Data>" can be strengthened to "<object: Data <associated_with: Student>>". In general, a requirement can be strengthened by adding slot-description pair(s), or by strengthening a description. Weakening is the converse of strengthening. The notion of strengthening or weakening requirements maps elegantly into the notion of subsumption in DL and is supported by off-the-shelf reasoners for a subset of our language.

Our main objective is to empirically evaluate *Desiree*, thereby answering the key question: Does *Desiree* improve the RE process? In particular, the paper presents three empirical studies involving upper-level software engineering students through controlled experiments. The results provide strong evidence that with a training time of two hours, *Desiree* can indeed help people to identify and address issues when refining stakeholder requirements. This work builds on our earlier publications that introduced our language for requirements and presented most of the operators [12][18][19].

The remainder of the paper is structured as follows. Section 2 introduces *Desiree*, Section 3 describes the three experiments, and section 4 presents and discusses the results. Section 5 discusses related work, while Section 6 concludes, and sketches directions for further research.

2 The *Desiree* Framework

In this section, we present the *Desiree* framework [18], including a set of requirement concepts, a set of requirement operators, a description-based syntax for representing these concepts and operators, and a systematic methodology for transforming stakeholder requirements into a requirements specification.

2.1 Requirements Concepts

The core notions of *Desiree* are shown in Fig. 1(shaded in the UML model). As in goal-oriented RE, we model stakeholder requirements as goals. We have 3 goal sub-kinds, 4

sub-kinds of specification elements (those with stereotype Specification Element), and domain assumptions, all of which are subclasses of Desiree Element. These concepts are derived from our experiences in analyzing the large PROMISE [22] requirements dataset. We use examples from this set to illustrate each of these concepts and relations.

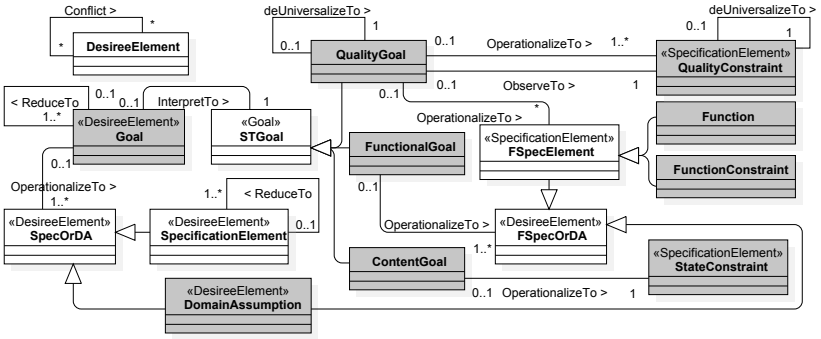

Fig. 1. The requirements ontology (adapted from [18])

There are three points to be noted. First, '>' and '<' are used to indicate the reading directions of the relations. Second, the relations (except "Conflict") are derived from applications of requirements operators (to be discussed in Section 2.2): if an operator is applied to an X object then the result will be *n..m* Y objects; if that operator is not applied, then there is no relation (thus we have 0..1 as the lower bound). Third, "STGoal", "SpecOrDA", "FSpecOrDA" and "FSpecElement" are artificial concepts, used to represent the union of their sub-classes, e.g., "STGoal" represents the union of "FunctionalGoal", "QualityGoal" and "ContentGoal". These classes are added to overcome the limitations of UML in representing inclusive (e.g., an operationalization of a functional goal) and exclusive (e.g., an interpretation of a goal) OR.

Functional Goal, Function and Functional Constraint. A functional goal (**FG**) represents a desired state, and can be operationalized by function(s), functional constraint(s), domain assumption(s), or a combination thereof. For example, the goal "student records be managed" specifies the desired state "managed". We capture the intention of something to be in a certain state (situation) by using the symbol ":<". So this example is modeled as an FG "Student_record :< Managed" (here "Managed" refers to an associated set of individuals that are in this specific state). This FG can be operationalized using functions such as "add", "update" and "remove" on student records.

When specifying a function (**F**), many pieces of information (e.g. actor, object, and trigger) can be associated with the desired capability. For example, "the system shall allow users to search products" will be captured as "F_0 := Search <subject: {the_system}><actor: User><object: Product>". A functional constraint (**FC**) constrains the situation under which a function can be manifested. As above, we specify intended situations using "$< s : D >$" pairs and constrain a function or an entity involved in a function description to be in such a situation using ":<". For example, "Only managers are able to activate debit cards" can be captured as "FC_1 := Activate <object: Debit_card> :< <actor: Manager>".

Quality Goal and Quality Constraint. We treat a quality as a mapping function that maps a subject to a value. A quality goal (**QG**) and quality constraint (**QC**) are

requirements that require a quality to have value(s) in a desired quality region (QRG) [12]. In general, a QG/QC has the form "Q (SubjT) :: QRG". For instance, "the file search function shall be fast" will be captured as a QG "Processing_time (File_search) :: Fast". When the subject consists of one or more individuals (instances, in object-oriented terms), we use curly brackets to indicate a set, e.g., "{the_system}". Note that QGs and QCs have the same syntax, but different kinds of QRGs: regions of QGs are vague (e.g., "low") while those of QCs are measurable (e.g., "[0, 30]").

Content Goal and State Constraint. A content goal (**CTG**) often specifies a set of properties of an entity in the real word. To satisfy a CTG, a system needs to be in a certain state, which represents the desired world state. That is, concerned properties of real world entities should be captured as data in the system. We use a state constraint (**SC**) to specify the desired system state. For example, to satisfy the CTG "a student shall have Id, name and GPA", the student record database table of the system must include three columns: Id, name and GPA. This example can be captured as "CTG_1 := Student :< <has_id: ID> <has_name: Name> <has_gpa: GPA>" and a state constraint (**SC**), "SC_2 := Student_record :< <ID: String> <Name: String> <GPA: Float>".

Domain Assumption. A domain assumption (**DA**) is an assumption about the operational environment of a system. For instance, "the system will have a functioning power supply", which will be captured as "{the_system} :< <has_ power: Power>". In *Desiree*, DAs are also used to capture domain knowledge, e.g., "Tomcat is a web server" will be captured as "Tomcat :< Web_server".

2.2 Requirements Operators

An overview of the requirements operators is shown in Table 1, where "#" indicates cardinality. As shown, *Desiree* includes two groups of operators (adapted from our previous work [18][19]): *refinement* and *operationalization*. In general, refinement operators refine goals to goals, or specification elements to specification elements, and operationalization operators map from goals to specification elements.

Table 1. An overview of the requirements operators

Requirements Operators		#InputSet	#OutputSet
Refinement	Reduce (R_d)	1	1..∗
	Interpret (I)	1	1
	deUniversalize (U)	1	1
	Resolve (R_s)	2..∗	0..∗
Operationalization	Operationalize (O_p)	1	1..∗
	Observe (O_b)	1	1

Reduce ($\boldsymbol{R_d}$). "Reduce" is used to refine a composite element (goal or specification element) to simple ones, or a high-level element to low-level ones. We allow making explicit domain assumptions when applying the "R_d" operator. For example, when reducing G_0 "pay for the book online" to G_1 "pay with credit card", one needs to assume DA_1 "having a credit card with enough credit". We capture this reduce refinement as "$\boldsymbol{R_d}$ (G_0) = $\{G_1, DA_3\}$".

Interpret ($\boldsymbol{I}$). "Interpret" generalizes the original "Disambiguate" operator, and allows us to not only disambiguate a requirement by choosing the intended meaning, but also classify a requirement and encode it using descriptions. For example, a goal G_1

"notify users with email" can be interpreted as a functional goal FG_2 "User :< Notified <means: Email>". We denote this as "$\boldsymbol{I}\ (G_1) = FG_2$".

deUniversalize ($\boldsymbol{U}$). $\boldsymbol{U}$ applies to a QG/QC to weaken the requirement, such that it is no longer expected to hold universallly. For example, going from "(all) file searches shall be fast", captured as QG1-1 "$Processing\_time\ (File\_search)\ ::\ Fast$", to "(at least) 80% of the searches shall be fast", captured as QG1-2 "$\boldsymbol{U}(QG1-1,\ File\_search,\ 80\%)$". Here we revised the signature of $\boldsymbol{U}$ by explicitly specifying the set of individuals to which $\boldsymbol{U}$ will be applied ("File_search", in this case).

Resolve ($\boldsymbol{R_s}$). In practice, some requirements will conflict with each other (i.e., they cannot be satisfied simultaneously). For example, the goal G_1 "use digital certificate" would conflict with G_2 "good usability" in a mobile payment scenario. We use a "*conflict*" relation to capture this phenomenon ("*Conflict* ($\{G_1, G_2\}$)"), and propose a new "Resolve" operator to address this conflict. In this example, we can replace G_2 by G_2' "acceptable usability" or drop G_1, which can be denoted as "$\boldsymbol{R_s}\ (\{G_1, G_2\}) = \{G_1, G_2'\}$" or "$\boldsymbol{R_s}\ (\{G_1, G_2\}) = \{G_2\}$", respectively.

Operationalize ($\boldsymbol{O_p}$). The $\boldsymbol{O_p}$ operator is used to operationalize goals into specification elements. In general, $\boldsymbol{O_p}$ takes as input one goal, and outputs one or more specification elements. For instance, the operationalization of FG_1 "Products :< Paid" as F_2 "Pay <means: Credit_card>" and DA_3 "Credit_card :< Having_enough_credit" will be written as "$\boldsymbol{O_p}\ (G_1) = \{F_2, DA_3\}$". One can use $\boldsymbol{O_p}$ to operationalize a QG as QC(s) to make it measurable, or as Fs and/or FCs to make it implementable. In addition, we can also operationalize a CTG as SC(s).

Observe ($\boldsymbol{O_b}$). The $\boldsymbol{O_b}$ operator is derived from the original "Agreement" operator by separating de-universalization from it. It is employed to specify the means, measurement instruments or human used to measure the satisfaction of QGs/QCs, as the value of slot "***observed_by***". Consider "(at least) 80% of the surveyed users shall report the interface is simple", which operationalizes and relaxes "the interface shall be simple". The original goal will be expressed as QG2-1 "$Style\ (\{the\_interface\})\ ::\ Simple$". To capture the relaxation, we first use $\boldsymbol{O_b}$, asking a set of surveyed users to observe QG2-1 and obtaining QC2-2 "$\boldsymbol{O_b}\ (QG2-1,\ Surveyed\_user)$", and then use $\boldsymbol{U}$, to require (at least) 80% of the users to agree that QG2-1 hold, i.e., "$\boldsymbol{U}\ (QC2-2,\ Surveyed\_user,\ 80\%)$".

2.3 A Transformation Methodology

The *Desiree* transformation process takes as input informal stakeholder requirements, and outputs a formal and consistent specification through incremental applications of the operators. We use a simple requirement "The system shall collect real time traffic info" to illustrate our three-staged *Desiree* method. The outputs of all stages are shown together in Fig. 2.

The informal stage. We first capture this requirement as a goal G_0. We then identify its concerns by asking "what does it concern?": a function "collect", a quality "timeliness" of collected traffic info, and a content concern "traffic info", and accordingly reduce G_0 to G_1, G_2 and G_3.

The interpretation stage. At this stage, we interpret G_1 to a functional goal FG_4, G_2 to a quality goal QG_7, G_3 to a content goal CTG_9, and encode the derived goals using our description-based syntax.

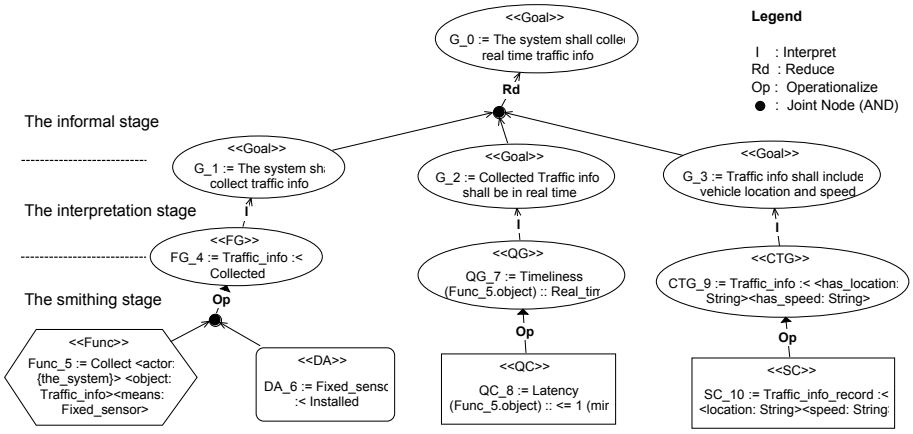

Fig. 2. An illustrative example for the *Desiree* method (with stereotypes on nodes)

The smithing stage. At this stage, we operationalize the structured goals into specification elements. For example, we operationalize FG_4 "Traffic_info :< Collected" as a function "$Func_5$:= Collect <actor: {the_system}> <object: Traffic_info> <means: Fixed_sensor>" and a domain assumption "DA_6 := Fixed_sensor :< Installed".

We have developed a prototype tool[1] to support the *Desiree* framework. Once we have derived a specification from stakeholder requirements, we can automatically translate it into a DL ontology using the prototype tool, and perform some useful reasoning tasks. For example, we can check which requirements are related to "Traffic_info" with the query "< relate_to: Traffic_info>".

Interested readers are referred to our previous work [18][19], where we have assessed the coverage of our requirements ontology by applying it to all the 625 requirements in the PROMISE dataset [22], evaluated the expressiveness of our description-based language by using it to rewrite all the 625 requirements in that dataset, and illustrated our methodology and available reasoning power by applying *Desiree* to a realistic *Meeting Scheduler* case study.

3 Experiments

In this section, we describe three controlled experiments, conducted to assess whether *Desiree* can indeed help people to conduct better requirement analysis. In the experiments, we compared *Desiree* with a *Vanilla* RE approach, where a participant uses the characteristics of a good software requirement specification (SRS) adapted from the IEEE standard 830 [8] and a set of guidelines for writing good requirements introduced in Wiegers et al. [23]. In the *Vanilla* method, participants manually go through and improve stakeholder requirements using these desirable characteristics and guidelines. This process approximates requirements walkthroughs using inspection checklists, and is used as a baseline representing how requirements are improved in practice.

To prepare, we defined a set of requirements issues as in Table 2 by identifying the inverse of each characteristic introduced in the IEEE standard [8]. Our experiments

[1] The tool is available at https://goo.gl/oeJ9Fi.

check to see if people can identify more of these issues when refining stakeholder requirements with *Desiree* or with the *Vanilla* RE approach. We do not consider the "Ranked" characteristic, as *Desiree* currently does not support requirements prioritization. We also do not compare *Desiree* with the *Vanilla* method on "Traceability" because *Desiree* is a goal-oriented method, and as such it intrinsically supports requirements to requirements traceability (requirements to sources, and requirements to design traceability are out of scope for our experiments). In addition, we introduce "Unsatisfiable", which is practically important but missing in the IEEE standard.

Table 2. Requirements issues

Issue	**Definition**
Invalid	A requirement is invalid if it is not the one that stakeholders want.
Incomplete	(1) incomplete requirement - a requirement is incomplete if necessary information is missing for implementation; (2) incomplete specification: an SRS is incomplete if any requirement is missing.
Ambiguous	A requirement is ambiguous if it has more than one interpretation.
Unverifiable	A requirement is unverifiable if it specifies unclear or imprecise value regions.
Inconsistent	An SRS is inconsistent if there are: (1) conflicts between requirements; (2) terms are used in different ways in different places.
Unmodifiable	An SRS is un-modifiable if its requirements are : (1) not structurally organized; (2) redundant; or (3) intermixing several requirements
Unsatisfiable	A requirement is practically unsatisfiable (attainable) if it is impossible or too costly to fulfill.

3.1 Research Question

Our research question can be stated as: *compared with the Vanilla method, can Desiree help people to identify more requirements issues when transforming stakeholder requirements to specifications*?

We define the null hypothesis, H_0, as: there is no statistical difference in the number of requirements issues found when using *Desiree* (μ_D) vs. the *Vanilla* method (μ_V). The alternative hypothesis, H_1, is accordingly defined as: there is a positive statistical difference in the number of issues found using *Desiree* vs. the *Vanilla* method. These two hypotheses can be formulated as Eq. 1. Similarly, we can define the null and alternative hypotheses for all the 7 kind of issues defined in Table 2. Due to space limitation, we do not present them here.

$$\begin{aligned} H_0 &: \mu_D - \mu_V = 0; \\ H_1 &: \mu_D - \mu_V > 0; \end{aligned} \tag{1}$$

3.2 Experiment Design

The experimental task was to transform a set of given stakeholder requirements into a specification through refinements. To evaluate the differences in their performance, each participant was required to perform the task twice: s/he uses the *Vanilla* method on a project X in the first session, and then uses *Desiree* on another project Y in the second session. In both sessions, the participants discussed with stakeholders to elicit necessary information for addressing identified issues (e.g., one probably needs further information to quantify "fast", which is vague), and submitted a refined specification. All experimental tasks were performed electronically and online.

The experiment was duplicated three times, the first at University of Trento, Italy, and the second and the third at Tsinghua University, China. In each experiment, we

used two projects, *Meeting Scheduler* (MS) and *Realtor Buddy* (RB), which are selected from the PROMISE requirements set [22], for the experimental task. We chose 10 requirements, which cover some typical functionalities and qualities (e.g., search, usability), from each project, and identified a list of issues for both projects. We also added some issues that are newly identified by participants into the reference issue lists in each experiment. Roughly, each project has approximately 45 issues, and each issue type has around 5 instances (except incomplete, which accounts for nearly 50% of the issues in each project). The statistics of these issues are available at https://goo.gl/oeJ9Fi.

In experiment one, we had 17 participants: Master's students at the Department of Information Engineering and Computer Science, University of Trento, taking the RE course at the spring term of 2015. We also had 4 Ph.D. students or postdocs in the research group of Software Engineering and Formal Methods playing the role of stakeholder. We assigned two stakeholders to a project, and randomly separated the students into two teams, RG1 and RG2. In the *Vanilla* session, we introduced the characteristics of a good SRS [8] and the textbook techniques [23] in 30 minutes, presented the domain knowledge of the two projects in 10 minutes, and tested in 90 minutes. In this session, RG1 worked on MS and RG2 worked on RB. In the *Desiree* session, we introduced the framework and its supporting tool in 40 minutes, and tested in 90 minutes. In this session, the teams were given the other project.

In experiment two, we had 18 volunteer participants: Master's students at the Institute of Information System and Engineering, School of Software, Tsinghua University. Compared with experiment one, a few changes were made to the experimental design. First, to improve the consistency of stakeholders' answers, we randomly separated the 18 participants into 6 small teams of size $2-4$, and hired 1 constant stakeholder for all the 6 teams on the same project (the 6 teams conducted the experiment one by one, not concurrently). Second, based on our initial observations that the training time was too short, we increased the *Desiree* training time from 40 minutes to 2 hours, 1 hour for the method and 1 hour for the tool (we also increased the *Vanilla* training time to 1 hour). In addition, we had updated the *Desiree* tool based on the feedback collected in experiment one, mainly on the usability aspect (e.g., copy and paste).

In experiment three, we had 30 participants [2]: Master's students at the School of Software, Tsinghua University, taking the RE course at the fall term of 2015. This experiment replicates the first, with the only change to *Desiree* training time: 45 minutes for the method, and 60 minutes for the tool (the *Vanilla* training time is 45 minutes). Also, we hired 6 students who have already participated experiment two as our stakeholders (7 in total, including the trainer). Similarly, stakeholders are randomly assigned to the two projects, and students are randomly separated into two teams. The two teams were given different projects in the two sessions.

4 Results

In this section, we report and discuss the experiment results. The data statistics collected from the three experiments are shown in Table 3. There are three points to be noted.

[2] We conducted experiment three at Tsinghua University again because: (1) the sample size of experiment two (15) is relatively small; (2) we had additional available participants at Tsinghua (one of the authors was teaching a RE class in the fall term there).

First, in each session of the three experiments, participants were expected to discuss questions over the given requirements via text interface with stakeholders, and produce a refined textual requirements specification or a refined *Desiree* requirements model. Second, in each experiment, the output of the *Vanilla* session was only text while that of the *Desiree* session could be a mix of models and texts: if a participant cannot model all the given requirements using the *Desiree* syntax and tool within specified time, s/he was required to refine unmodelled requirements using natural language, but still following the *Desiree* method. In a few cases, participants submitted only models or only textual specifications in a *Desiree* session. Third, in the experiments, a few participants (e.g., 2 in the *Vanilla* session of experiment three) have refined the given requirements without discussing any questions with stakeholders.

Table 3. Statistics of collected data: conversations, requirements texts and models

	Experiment One		**Experiment Two**		**Experiment Three**	
Session	*Vanilla*	*Desiree*	*Vanilla*	*Desiree*	*Vanilla*	*Desiree*
Time (*min*)	63	89.5	74	94	62	97
Discussions	17	16	18	15	28	29
Textual Requirement Specifications	17	14	18	12	30	23
Desiree Requirement Models	-	11	-	15	-	29
Complete Samples	16		15		29	

4.1 Descriptive Statistics

We carefully went through participants' discussions and refined requirements (both texts and models) to check how many issues they have identified in the experiments. We say a participant has identified an issue if either of the two conditions hold.

1. A participant has asked a corresponding question, e.g., we gave a count of identified unverifiable issue if someone has asked "how to measure fast?" to quantify "fast".
2. A participant has eliminated an issue in his/her refined requirements specification, either texts or models, although s/he did not ask any related question. E.g., a participant has eliminated a term inconsistency in RB by changing "the product" to "the system" without asking any questions.

To keep consistency, the trainer performed the evaluation for all the three experiments. We show the average percentage of identified issues of participants in Table 4, where positive results are in bold. We see that in experiment one, on average, a participant was able to find more issues with *Desiree* than with the *Vanilla* method (35.79% vs. 30.08%), but discovered fewer "Ambiguous" and "Unverifiable" issues. In experiment two, as the training time for *Desiree* increased from 40 minutes to 2 hours, we can see that the participants performed better in general: they found more issues in total (46.27% vs. 32.93%). Experiment three has provided similar evidence as experiment two: with a training time of near 2 hours, participants are able to find more issues with *Desiree* (39.98% vs. 28.17%). In addition, we have compared the performance of each participant on the two sample projects (MS and RB). The detailed statistics (including raw data and statistical calculations) are available at https://goo.gl/oeJ9Fi.

4.2 Hypothesis Testing

We statistically analyzed the participants' differences in terms of identified issues when using *Desiree* vs. the *Vanilla* approach. Since we have far less than 30 participants in

Table 4. Statistics of issues identified by participants in the three experiments

	Experiment One			**Experiment Two**			**Experiment Three**		
	Vanilla	*Desiree*	Diff	*Vanilla*	*Desiree*	Diff	*Vanilla*	*Desiree*	Diff
Incomplete	15.84%	19.70%	**3.86%**	27.30%	31.64%	**4.34%**	21.49%	30.77%	**9.28%**
Ambiguous	24.22%	20.31%	-3.91%	32.22%	54.44%	**22.22%**	12.93%	38.45%	**25.52%**
Inconsistent	10.42%	15.62%	**5.20%**	6.67%	8.89%	**2.22%**	16.09%	16.38%	**0.29%**
Unverifiable	88.75%	84.37%	-4.38%	81.33%	92.67%	**11.34%**	79.37%	90.27%	**10.90%**
Unmodifiable	20.83%	47.92%	**27.09%**	9.44%	43.33%	**33.89%**	3.45%	47.41%	**43.96%**
Unsatisfiable	2.78%	12.44%	**9.66%**	10.56%	50.00%	**39.44%**	3.45%	24.14%	**20.69%**
Total	30.08%	35.79%	**5.71%**	32.93%	46.27%	**13.34%**	28.17%	39.98%	**11.81%**

experiment one and two, and our Shapiro-Wilk Normality tests showed that the participants' differences in experiment three are not normally distributed on 3/7 of the issue indicators, we employed both *paired Student's t* test [2] and *Wilcoxon Signed-Rank* test (WSR) [3] for our one-tailed hypothesis testing. The paired T test assumes that the differences between pairs (repeated measurements on a sample, before and after a treatment) are normally distributed, and is robust to moderate violation of normality [21]. As a complement, the WSR test is a non-parametric alternative to the paired T test if the differences between pairs are severely non-normal [21].

Table 5. Statistical p-value for issues identified by participants

	Experiment One		**Experiment Two**		**Experiment Three**	
	Paired T	**WSR**	**Paired T**	**WSR**	**Paired T**	**WSR**
Incomplete	0.08331	**0.03717**	0.10593	0.1221	**0.00007**	**0.00021**
Ambiguous	0.71236	0.6753	**0.00144**	**0.00288**	**0.00069**	**0.00145**
Inconsistent	0.37845	0.37097	0.33507	0.30345	0.47585	0.66204
Unverifiable	0.92169	0.8449	**0.01428**	**0.02747**	**0.00396**	**0.00519**
Unmodifiable	**0.00269**	**0.003**	**0.00001**	**0.00052**	<**0.00001**	<**0.00001**
Unsatisfiable	**0.04449**	0.05155	<**0.00001**	**0.00032**	**0.00006**	**0.00013**
Total	0.076	0.06023	<**0.00001**	**0.00036**	<**0.00001**	**0.00001**

We report the p-values in Table 5. We can see that there is strong evidence that *Desiree* can help people to identify more issues in general (the last row): for both tests, p-value $\leq 0.00036 << \alpha = 0.05$ (the common confidence level) in experiment two and three, and p-value ≈ 0.05 in experiment one. Specifically, there are strong evidence that *Desiree* is able to help people to identify more "Incomplete", "Ambiguous", "Unverifiable", "'Unmodifiable', and "Unsatisfiable" issues (their p-values ≤ 0.05 in at least two experiments). We also see that there is no evidence that *Desiree* can help people to identify more "Inconsistent" issues (p-value $>> 0.05$) in all the three experiments.

To mitigate the potential risk of *accumulated type I error* (a false rejection of the null hypothesis due merely to random sampling variation) when running multiple tests [17], we applied the Bonferroni adjustment [1] to the p-values obtained in experiment three. We report the adjusted p-values and the related statistics (i.e., t values for the paired T tests, Z values for the WSR tests) in Table 6. The very small adjusted p-value (p-value $< 0.00001 << 0.05$) indicates a very strong evidence that the samples are not from the null distribution. We can hence reject the null hypothesis H_0 stated in Eq. 1 at the confidence level $\alpha = 0.05$, and accept the alternative hypothesis H_1.

We also analyzed the *effect sizes*, which are shown Table 6. Effect size is the magnitude of a treatment effect [6], i.e., it tells to what degree a treatment (e.g., the *Desiree*

Table 6. Analyzing experiment three: p-values, statistics, and effect

	Paired T Test				**Wilcoxon Signed-Rank (WSR)**			
	Pvalue	Bonferroni	t(28)	Effect	Pvalue	Bonferroni	Zvalue	Effect
Incomplete	0.00007	0.00046	4.4302	0.82266	0.00021	0.0015	3.52263	0.64314
Ambiguous	0.00069	0.0048	3.5536	0.6599	0.00145	0.01017	2.97754	0.54362
Inconsistent	0.47585	1	0.06111	0.01135	0.66204	1	-0.41805	-0.07633
Unverifiable	0.00396	0.02773	2.8598	0.53105	0.00519	0.03631	2.5631	0.46796
Unmodifiable	<0.00001	<0.00001	8.6828	1.61236	<0.00001	0.00002	4.57097	0.83454
Unsatisfiable	0.00006	0.00044	4.4458	0.82556	0.00013	0.00094	3.64313	0.66514
Total	<0.00001	<0.00001	6.5681	1.21967	0.00001	0.00008	4.24921	0.7758

method) affects the participants. For example, according to Coe [6], an effect size of 0.8 means that the score of the average person in the experimental group is 0.8 standard deviations above the average person in the control group, and hence exceeds the scores of 79% of the control group. We checked the effect sizes in experiment three for each kind of requirements issue [3], and found that this interpretation matches very well with the actual situation. Using Cohen's conventional criteria of "small" (effect size from 0.2 to 0.3), "medium" (around 0.5), or "big" (0.8 to infinity) effect [7], the effect sizes for "Total", "Incomplete", "Ambiguous", "Unverifiable", "Unmodifiable" and "Unsatisfiable" issues in experiment three fall into the"medium" or "large" category.

4.3 Analysis

In general, the results meet our expectations.

Incomplete. In our observations, *Desiree* is helpful in identifying incomplete requirements issues mainly because: (1) the description-based syntax drives users to think about the kinds of properties that shall be associated with the capability when specifying a function; (2) the syntax facilitates the consideration of "which attributes shall be used to describe the description (filler)?" when specifying a slot-description pair. Take "the system shall be able to search meeting rooms records" as an example, with *Desiree*, many participants were able to find the following missing information: who can search? what kinds of search parameters shall be used? Further, more participants have asked "what kinds of information shall a meeting room record include?", identifying a missing content requirement.

Ambiguous. *Desiree* offers operational rules for identifying potential ambiguities: (1) checking the subject of a slot (property); (2) checking the cardinality of the restriction of a slot in a function description. These rules are shown to be useful in our experiments. For example, more participants have identified the ambiguity in the requirement "the system shall be able to download contact info for client": is "for client" attached to the function "download" or the entity "contact info"? More interestingly, for the requirement "the system shall allow privileged users to view meeting schedules in multiple reporting views", after addressing the unverifiable issues of "privileged user" and "multiple", several participants have further asked "Shall these reporting views be opened simultaneously or not?", identifying an implicit ambiguity issue.

Unverifiable. We observed that the participants can easily find simple unverifiable issues in given requirements, but tend to miss "deep" vague issues in stakeholders' answers when using the *Vanilla* method. With *Desiree*, the structuring of each requirement

[3] The effect sizes in experiment one and two can be found at https://goo.gl/oeJ9Fi.

could remind them about implicit unverifiable issues. For example, most of the participants were able to justify "the product shall have good usability" as unverifiable, but few of them realized that "the product shall be easy to learn for realtors", which was given by stakeholders as a refinement of the previous requirement, is still vague. With *Desireee*, participants would keep asking "how to measure easy?". That is, when using the *Vanilla* method, participants were more likely to accept vague stakeholder answers, while using *Desiree*, they were more likely to notice and correct vague responses.

Un-modifiable. *Desiree* requires users to identify the concerns of a requirement, and separate them if there are several. This helps to avoid intermixed requirements. With *Desiree*, many participants were able to successfully decouple composite requirements into simple ones. For example, they decoupled "the system shall be able to generate a CMA (Comparative Market Analysis) report in acceptable time" into "generate a CMA report" (F_1 := Generate <object: CMA_report>) and "the generation shall be in acceptable time" (QG_2 := Processing_time (F_1) :: Acceptable). Further, they were able to capture interrelations between requirements by utilizing the *Desiree* tool. For example, in the above example, the two elements are interrelated through the use of F_1 as the subject of QG_2. This enables us to systematically identifying the requirements to be affected when updating a requirement.

Un-satisfiable. *Desiree* offers a "de-Universalize" operator for weakening requirements in order to make them practically satisfiable. The supporting tool also provides hints for relaxation when the "Observe" operator is applied. As such, the participants were able to identify more potentially un-satisfiable issues. For example, when operationalizing the QG "the search of meeting rooms shall be intuitive" by assigning surveyed users, many of them have asked "how many percentage of the surveyed users shall agree?"

Inconsistent. Our framework assumes that conflicts are explicitly defined by analysts and provides an "Resolve" operator to resolve them. As such, the framework does not as yet offer much help in identifying inconsistency issues.

We had some additional observations over experiment results. First, we omitted "Invalid" from our experiment results since there were no invalid issues in the two projects (it is hard to justify which requirement is not desired by original stakeholders who provided the requirements set). Second, the participants in experiment one had a poorer performance on identifying "Ambiguous" and "Unverifiable" issues mainly because the training time of 40 minutes is too short: many students have spent a lot of time struggling with the syntax and the tool, and did not have enough time to analyze the requirements themselves. Third, the learning of *Desiree* varies from individual to individual: in experiment three, 24 out of the 29 participants (82.76%) have better performance when using the *Desiree* method while the rest (5/29, 17.24%) have slightly poorer performance.

4.4 Feedback

We have conducted a survey on the two RE methods in each experiment. In general, the majority of the participants have reported that the *Desiree* framework is useful or very useful, but hard to learn. For example, among the 24 collected responses in the survey of experiment three, 20 out of 24 (20/24) have reported that *Desiree* is useful or very

useful, and 11 out of them (11/24) have reported that *Desiree* is hard to learn. Specifically, the participants have pointed out that the framework is useful because it offers a structured way for classifying and representing requirements, and provides a systematic method for reducing complex requirements; it is hard to learn mainly because of its grammar. We have also got positive feedback from the participants in each experiment. Interested readers are referred to our survey reports available at https://goo.gl/oeJ9Fi.

1. "*Desiree* embodies correctness check. It enforces you to think if what you are doing is right, e.g., functional goals, quality goals ...."
2. "The method helps a lot when reducing the complex requirements and help with the standard representation of those items. Nothing is useless. The method makes the analysis process clearer more or less. "
3. "The tool makes me thinking in the structural and the mind is more MECE (Mutually Exclusive, Collectively Exhaustive)."

4.5 Threats to Validity

There are several threats to the validity of our evaluation.

Independence between participants. We have tried to minimize mutual interference between participants in each experiment by: (1) assuming an exam scenario and asking them to perform the experimental task individually; and (2) requiring them to use text to communicate with stakeholders instead of speaking aloud (we had only 4 face-to-face conversations in the *Vanilla* session of experiment three since these students did not bring their laptops).

Assessment. The experiment results were evaluated by only one person. We have used objective and consistent rules for making judgments, to minimize the impact of individual subjectivity.

The nature of participants. Most of the participants in our experiments are students. However, in experiment three, at least 8 participants (8/29) have more than 1 year work experience (5/8 specific to RE). Also, holding the studies in two different universities provides more confidence in the generalizability of our results. We could further minimize this threat by conducting experiments in an realistic industrial setting.

Order. *Desiree* is used after the *Vanilla* method in each experiment. Although each participant applied *Desiree* to a different project in the second task, s/he may have learned from the *Vanilla* application in her/his first task. We could have done counterbalancing: some groups apply *Vanilla* then *Desiree*, and others apply *Desiree* then *Vanilla*; however, this setup would have been difficult to implement as part of the course design, with alternating tutorials and exercises for different groups of students.

Sample size. The sample size of 29 in experiment three is sufficient to assume the normality for paired T test. Also, we have ran the Wilcoxon Signed-Rank test, which does not assume any distribution of the population, in each experiment. The threat of generalizing our conclusion is relative low.

Training. In our experiments, the *Desiree* framework was taught by the designer; and how the *Vanilla* RE method was taught may affect results. We have tried to be fair in teaching and not bias the results.

Projects. The *Desiree* method can be more or less successful for different types of projects, e.g., larger or more realistic. We have tried to mitigate this by using more than one project.

5 Related Work

In the RE literature, many empirical evaluations have been conducted to assess the utility of some languages or methods, but mainly on their expressiveness and effectiveness [10][13]. Al-Subaie et al. [4] have used a realistic case study to evaluate KAOS and its supporting tool, Objectiver, with regarding to a set of properties of requirements, introduced in Davis et al. [9] and the IEEE Std 830-1998 [8]. They reported that KAOS is helpful in detecting ambiguity and capture traceability, but the formalism of KAOS is only applicable to goals that are in enough detail and can be directly formalized.

Work by Matulevicius et al. [20] is quite relevant. In their evaluation, the authors have compared *i** and KAOS through experiments. Besides the quality of languages themselves, they also compared the models generated by using the two frameworks with regarding to a set of qualitative properties in the semiotic quality framework [16]. Their findings indicate a higher quality of the KAOS language (not significant), but a higher quality of the *i** models (the participants are not required to write formal specifications with KAOS). They also found that the goal models produced by both frameworks are evaluated low at several aspects, including verifiability, completeness and ambiguity.

These evaluations show that requirements initially captured in goal models are of low quality and error prone, and techniques for incrementally improving the quality of requirements captured in traditional goal models are needed. We have proposed *Desiree* for addressing such deficiencies [18][19], in this work we conducted three experiments to evaluate its effectiveness.

6 Conclusion

In this paper, we have presented a series of three controlled experiments that are conducted to evaluate the effectiveness of the *Desiree* framework. The evaluation results have provided strong evidence that given a training time around two hours, *Desiree* indeed can help people to perform better requirements analysis (e.g., less ambiguous, more complete, etc.) with a medium or big effect.

There are several directions can be further explored. First, the usability of the *Desiree* tool, and the accessability of the tutorial for the *Desiree* approach (e.g., wiki, video, help manual) needs to be improved. Second, our *Desiree* framework currently does not have a built-in set of slots, and may result in different outputs when used by different users as they may use different words for the same relation (e.g., “belong to” vs. “associated with”). An interesting idea is “slot mining”: statistically analyzing the requirements in specific application domains and eliciting a set of frequent slots.

Acknowledgements. This research has been funded by the ERC advanced grant 267856 “Lucretius: Foundations for Software Evolution” (April 2011 - March 2016). It has also been supported by the Key Project of National Natural Science Foundation of China (no. 61432020), and the Key Project in the National Science & Technology Pillar Program during the Twelfth Five-year Plan Period (No. 2015BAH14F02). Jennifer is supported by an ERC Marie Skodowska-Curie Intra European Fellow-ship (PIEFGA - 2013 - 627489) and by a Natural Sciences and Engineering Research Council of Canada Postdoctoral Fellowship (September 2014 - August 2016).

References

1. Bonferroni correction (Nov 2015), https://en.wikipedia.org/w/index.php?title=Bonferroni_correction&oldid=692500900
2. Student's t-test (Oct 2015), https://en.wikipedia.org/w/index.php?title=Student%27s_t-test&oldid=687517571
3. Wilcoxon signed-rank test (Nov 2015), https://en.wikipedia.org/w/index.php?title=Wilcoxon_signed-rank_test&oldid=690943842
4. Al-Subaie, H.S., Maibaum, T.S.: Evaluating the effectiveness of a goal-oriented requirements engineering method. In: CERE'06. pp. 8–19. IEEE (2006)
5. Baader, F.: The description logic handbook: theory, implementation, and applications. Cambridge university press (2003)
6. Coe, R.: It's the effect size, stupid: What effect size is and why it is important (2002)
7. Cohen, J.: Statistical power analysis for the behavioral sciences. Academic press (2013)
8. Committee, I.C.S.S.E.S., Board, I.S.S.: IEEE Recommended Practice for Software Requirements Specifications. IEEE (1998)
9. Davis, A.M.: Software requirements: objects, functions, and states. Prentice-Hall, Inc. (1993)
10. Estrada, H., Rebollar, A.M., Pastor, O., Mylopoulos, J.: An empirical evaluation of the i* framework in a model-based software generation environment. In: CAiSE'06. pp. 513–527. Springer (2006)
11. Fabbrini, F., Fusani, M., Gnesi, S., Lami, G.: The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool. In: IEEE/NASA SEW-26. pp. 97–105. IEEE (2001)
12. Guizzardi, R., Li, F.L., Borgida, A., Guizzardi, G., Horkoff, J., Mylopoulos, J.: An ontological interpretation of non-functional requirements. In: FOIS'14. vol. 267, pp. 344–357. IOS Press (2014)
13. Horkoff, J., Aydemir, F.B., Li, F.L., Li, T., Mylopoulos, J.: Evaluating Modeling Languages: An Example from the Requirements Domain. In: ER'14, pp. 260–274. Springer (2014)
14. Jackson, M., Zave, P.: Deriving specifications from requirements: an example. In: ICSE'95. pp. 15–24. ACM (1995)
15. Kamalrudin, M., Hosking, J., Grundy, J.: Improving requirements quality using essential use case interaction patterns. In: ICSE'11. pp. 531–540. ACM (2011)
16. Krogstie, J.: A semiotic approach to quality in requirements specifications. Proceedings of the IFIP TC8/WG8 1, 231–249 (2002)
17. LeBlanc, D.C.: Statistics: concepts and applications for science, vol. 2. Jones & Bartlett Learning (2004)
18. Li, F.L., Horkoff, J., Borgida, A., Guizzardi, G., Liu, L., Mylopoulos, J.: From Stakeholder Requirements to Formal Specifications Through Refinement. In: REFSQ'15, pp. 164–180. Springer (2015)
19. Li, F.L., Horkoff, J., Mylopoulos, J., Guizzardi, R.S., Guizzardi, G., Borgida, A., Liu, L.: Non-functional requirements as qualities, with a spice of ontology. In: RE'14. pp. 293–302. IEEE (2014)
20. Matuleviius, R., Heymans, P.: Comparing goal modelling languages: An experiment. In: REFSQ'07, pp. 18–32. Springer (2007)
21. McDonald, J.H.: Handbook of biological statistics, vol. 2. Sparky House Publishing Baltimore, MD (2009)
22. Menzies, T., Caglayan, B., He, Z., Kocaguneli, E., Krall, J., Peters, F., Turhan, B.: The PROMISE Repository of empirical software engineering data (Jun 2012)
23. Wiegers, K., Beatty, J.: Software requirements. Pearson Education (2013)