# Effects of Using Examples on Structural Model Comprehension

## A Controlled Experiment

Dina Zayan, Michał Antkiewicz, Krzysztof Czarnecki
Generative Software Development Lab, University of Waterloo, Canada, http://gsd.uwaterloo.ca
{dzayan, mantkiew, kczarnec}@gsd.uwaterloo.ca

## ABSTRACT

We present a controlled experiment for the empirical evaluation of Example-Driven Modeling (EDM), an approach that systematically uses examples for model comprehension and domain knowledge transfer. We conducted the experiment with 26 graduate and undergraduate students from electrical and computer engineering (ECE), computer science (CS), and software engineering (SE) programs at the University of Waterloo. The experiment involves a domain model, with UML class diagrams representing the domain abstractions and UML object diagrams representing examples of using these abstractions. The goal is to provide empirical evidence of the effects of suitable examples in model comprehension, compared to having model abstractions only, by having the participants perform model comprehension tasks. Our results show that EDM is superior to having model abstractions only, with an improvement of 39% for diagram completeness, 30% for questions completeness, 71% for efficiency, and a reduction of 80% for the number of mistakes. We provide qualitative results showing that participants receiving model abstractions augmented with examples experienced lower perceived difficulty in performing the comprehension tasks, higher perceived confidence in their tasks' solutions, and asked fewer clarifying domain questions, a reduction of 90%. We also present participants' feedback regarding the usefulness of the provided examples, their number and types, as well as, the use of partial examples.

## Categories and Subject Descriptors

D.2 [**Software**]: Software Engineering; D.2.1 [**Requirements/Specifications**]: Elicitation methods; D.2.1 [**Requirements/Specifications**]: Methodologies; D.2.13 [**Reusable Software**]: Domain engineering; I.6.5 [**Model Development**]: Modeling methodologies

## General Terms

Documentation, human factors


*ICSE* '14, May 31-June 7, 2014, Hyderabad, India

http://dx.doi.org/10.1145/2568225.2568270 ...$15.00.

## Keywords

Structural domain model comprehension, controlled experiment, abstraction, example, example-driven modeling, EDM

## 1. INTRODUCTION

The process of knowledge transfer occurs in most activities of software development. In this paper, we consider the transfer of domain knowledge among project stakeholders.

The main challenge in domain knowledge transfer is that much of the knowledge is complicated in nature, difficult to articulate, and communicate among stakeholders who have different backgrounds and skills [29]. Subject Matter Experts (SMEs) usually assume that business analysts (BAs) possess the same level of knowledge and they are able to infer the business rules that are required to build a software system that addresses the problems of SMEs. The problem is worsened when the BAs mistakenly think that they understand the domain and they do not require further elicitation sessions. This contributes to the production of incomplete and/or incorrect requirements artifacts, which is one of the major reasons for the failure of software projects [23, 11].

To help with knowledge transfer, structural modeling can be used for articulating, capturing, and communicating domain knowledge [29]. However, the constructed models often only contain domain abstractions that cannot be easily validated by other stakeholders who usually are not modeling experts [14]. Based on research results in cognitive psychology and in software engineering, we propose that **explicit examples should be used together with the abstractions for effective domain knowledge transfer** [10].

The process of collecting, communicating, and verifying domain knowledge iteratively through examples is what we refer to as *example-driven modeling* (EDM) [10]. Since the process of domain knowledge transfer is not exclusive to the phase of creating models, EDM is of equal importance when creating or exchanging models. In EDM, the purpose of examples is to explore and learn about the domain during all phases of software development.

Examples are commonly used in software engineering. Test cases are examples of what the software should do. Traces are example executions of behavioral models used for model validation [13, 27]. Approaches such as test-driven development [19], behavior-driven development [25], story test driven development [29], all postulate that examples should be used for specifying software behavior. The importance of using examples in structural modeling, on the other hand, seems to be underestimated [30, 39]. For example, UML object diagrams, which represent examples of more abstract

class diagrams, are rarely used in practice [30, 39].

It is important to verify whether the use of examples offers significant benefits in structural modeling. It is also important to know which types of examples, the number of examples, and their presentation forms are best for improving the comprehension of the model and, consequently, of the domain. This paper presents a controlled user experiment targeting these questions. The objective is to provide the missing empirical evidence about the effects of using examples on structural model and domain comprehension.

The results of the experiment show that augmenting model abstractions with explicit examples improved model and domain comprehension by participants who were novices to the application domain. The highest score for diagram completeness (69.6%) achieved by the participants who received the model abstractions only is lower than the lowest score of those who received the abstractions together with examples (81.1%). On average, participants receiving the examples asked 1 instead of 10 domain questions (a reduction of 90%), had 5 times fewer mistakes per object diagram (a reduction of 80%), scored 1.5 times higher for the questions response completeness, and had nearly double the efficiency as compared to the control group. EDM group participants also experienced lower perceived difficulty of performing the comprehension tasks, while at the same time, higher confidence in their answers. We conjecture that these significant results could potentially translate into significant time and cost savings in industry; however, industrial evaluation remains future work.

**Structure of the paper.** Section 2 presents a modeling approach where abstractions and examples are equally important parts of a model. Section 3 describes the design of our experiment and details its operational phase. Section 4 describes our data collection techniques. Section 5 presents our preliminary data and subjects analysis. Section 6 discusses our results and major findings, followed by a presentation of the threats to validity in Section 7. Concluding remarks and future work are in Section 9.

## 2. EXAMPLE-DRIVEN MODELING

In order to better understand and judge our experimental design and make the implications of our hypotheses testing more apparent [22], this section presents the theory from which our hypotheses are derived.

We have previously proposed example-driven modeling (EDM), an approach where modeling is based on examples [10]. EDM relies on the systematic use of examples for eliciting, validating, and verifying complex domain knowledge. The use of examples in current structural modeling practices, at least in the UML context, is undervalued [30, 39]. This limits the use of models to highly trained experts [14] who are able to work with and understand abstractions. However, many stakeholders who would benefit from the models are not experts. Yet, they can understand and prefer working with models via examples [14, 18]. According to Van God et. al. [36], the use of examples lowers the mental effort required to understand problems.

In practice, class diagrams are much more frequently used than object diagrams [30, 39]. This suggests that even if examples are used in the structural modeling world, they are used informally. In EDM, models comprise both abstractions and examples ("*model = abstractions + examples*"), where abstractions are representations that are disassociated from concrete objects, and the examples are concrete instances of these abstractions. We hypothesize that fostering model comprehension is achieved if model abstractions are augmented with examples, since humans experience optimum knowledge transfer when they learn abstractions together with examples [15, 17]. In this experiemnt, we address the following questions: *What is the impact of examples on model comprehension? How to use examples systematically? What kind of examples are more useful than others?*. We derive our hypotheses and attempt to answer these questions based on cognitive science theories.

The use of examples varies between novices and experts. While novices would benefit instantly from examples for comprehension, experts would only need examples for clarifications [21, 36, 12]. According to Kalyuga [21], examples help compensate for the missing or partially available information. Hence, the difference in the need of examples is attributed to the user's level of expertise in the domain. We hypothesize that BAs who are familiar with the domain prefer abstractions and seek examples only for clarifications. On the other hand, BAs who are new to the domain prefer to work with examples first to understand the abstractions.

The choice of examples is important for effective comprehension and domain knowledge transfer. While positive (valid) examples specify correct model instances, negative (invalid) examples specify undesired or unacceptable instances. Invalid examples can be abstracted as model constraints. According to Gick and Paterson [16], the most effective types of examples are *near-miss contrasting examples*. They help humans build abstractions by emphasizing the critical differences between the instances.

Partial examples are needed to allow expressing only the relevant information. Partial examples can be expressed using partial instances, whereby some elements (e.g., objects, attributes, links) can be ommitted or their presence can be uncertain; and whereby values can be unspecified (it is unknown whether a value is present or not) or partially specified (value is present but unknown) [8, 33]. We hypothesize that having partial examples limited to a given context is more beneficial than having full complete instances of the model. This way the novices to the domain can focus on a small subset of the domain at a time.

## 3. EXPERIMENTAL DESIGN

The purpose of our experiment is to quantitatively and qualitatively evaluate the effectiveness of EDM as a structural modeling approach when compared to a modeling approach whereby the model consists of abstractions only. We followed the guidelines for software engineering (SE) experimentation presented in [20, 22, 40]. The goal of the experiment is summarized using the GQM template [37] (Table 1).

### 3.1 Research Questions and Hypotheses

The research questions underlying our experiment are:

- **RQ1:** Does augmenting model abstractions with explicit examples improve model comprehension among model receptors who are novices to the application domain, compared to having model abstractions only?
- **RQ2:** Does using a variety of valid and invalid examples improve the model receptors' comprehension of the model abstractions, compared to having valid examples only?

**Table 1: Experimental goal according to GQM template.**

| |
|---|
| **Analyze** the effects of using examples on comprehension of structural models by model receptors<br>**for the purpose of** improving the comprehension<br>**with respect to** correctness, completeness, and efficiency of solving instantiation tasks and answering domain questions<br>**from the perspective of** the researcher who is the knowledge provider<br>**in the context of** knowledge transfer to model receptors who are novices to the domain. |

We use several measures of comprehension as dependent variables (in literature also referred to as "response variable") for running the experiment. Accordingly, we formulate the following *null*/**alternative** hypotheses:

$H1_0$/**H1**: Augmenting model abstractions with explicit examples [*does not impact*]$_0$/[**improves**] the completeness of the object diagram created by each participant.

$H2_0$/**H2**: Augmenting model abstractions with explicit examples [*does not impact*]$_0$/[**reduces**] the number of mistakes in a given participant's diagram.

$H3_0$/**H3**: Augmenting model abstractions with explicit examples [*does not impact*]$_0$/[**increases**] the number of the questions answered correctly by each participant.

$H4_0$/**H4**: Augmenting model abstractions with explicit examples [*does not impact*]$_0$/[**improves**] the participant's efficiency in solving the questions.

$H5_0$/**H5**: Augmenting model abstractions with explicit examples [*does not impact*]$_0$/[**reduces**] the participants' perceived difficulty in performing the comprehension tasks.

$H6_0$/**H6**: Augmenting model abstractions with explicit examples [*does not impact*]$_0$/[**increases**] the participants' perceived confidence in the comprehension tasks' solutions.

$H7_0$/**H7**: Having a variety of valid and invalid examples [*does not impact*]$_0$/[**improves**] model comprehension by the participants, compared to having valid examples only.

The last null and alternative hypotheses correspond to the second research question. We perform quantitative analysis to accept/reject the first four null and alternative hypotheses, while we depend on qualitative feedback from the participants for the last three null and alternative hypotheses.

## 3.2 Dependent and Independent Variables

The purpose of this experiment is to verify whether EDM improves model and, consequently, domain comprehension among stakeholders who are considered novices to the application domain, compared to a structural modeling approach where the model consists of abstractions only. Consequently, our experiment has one independent variable: the *modeling method* used to understand the model abstractions. The modeling method varies across two groups: the control group which gets a model consisting of abstractions only, and the EDM group which gets a model consisting of both abstractions and examples. The design of our experiment is a *between-subjects* design where each participant is either a part of the control group or the EDM group.

Similarly to other empirical evaluations of UML comprehension [20, 31, 32, 34, 35], the dependent variables of our experiment are *diagram completeness*, *diagram mistakes*, *questions response completeness*, the *efficiency* in answering the questions, the participants' *perceived difficulty* in performing the comprehension tasks, and finally the participants' *confidence* in their task solutions.

## 3.3 Problem Domain

We selected *rewards loyalty programs* as the problem domain of our experiment. Loyalty programs represent structured marketing activities, especially in the retail domain, that reward and therefore encourage loyal buying behavior at the customer end. The choice of rewards loyalty programs was inspired by a similar domain modeled and used by one of our industrial partners. The domain is sufficiently rich in concepts, relationships, and constraints, and we had enough knowledge to build model abstractions and examples.

## 3.4 Participants

In total, we had 28 participants in the experiment: 26 model receptors, 1 SME, and 1 model creator. For model receptors, we decided to target graduate and undergraduate students from electrical and computer engineering (ECE), computer science (CS), and software engineering (SE) programs at the University of Waterloo. The year of study for the undergraduate students varied from second year to fourth year. Most students had previous work experience either through full-time jobs or co-op terms, as well, as familiarity with UML either through academic projects or work experience. To avoid biasing the results through model receptors' expectations, we employed blinding techniques. The model receptors were not familiar with the experiment hypotheses nor with the measures we were applying. The SME and the model creator were the first and second authors of the paper. They were both familiar with entire experimental setting, and they had advanced knowledge of UML class and object diagrams.

## 3.5 Tasks and Treatments

EDM provides aid in the comprehension of model abstractions and the problem domain. Our approach supports BAs, designers, developers, and any role who needs to understand the domain for future use. Our aim was to design a set of tasks that are close to scope and complexity to real tasks performed by practitioners, and at the same time allow us to objectively measure the difference between the comprehension levels of the two modeling approaches. We asked the participants from both the control and EDM groups to complete the same set of tasks in a single three-hour session. Model receptors were randomly assigned to treatments.

The experiment involved two tasks. **Task 1** asked the participants to create an object diagram for a rewards loyalty program called Club Sobeys. We provided a brief overview about what Club Sobeys is. The instructions specified that the participants should create at least one instance of every class, include all associations and attribute values, and realize all requirements included in the experiment document. For example, one requirement was "*Provide information about the different sales channels through which the customer will know about the offers. Take into account different ways of promoting store and partner offers.*" We provided the participants with a checklist that included all the main concepts they needed to include in their object diagrams: member registration, points accumulation through in-store

products purchased without offers, points accumulation for in-store offers, sales channels to promote all offers, partner offers, and redemption mechanisms. The control group received the model abstractions (class diagram) in this task. However, the EDM group received both the model abstractions and the six partial examples for a different rewards loyalty program called Shoppers Optimum.

**Task 2** of the experiment asked the model receptors to answer a set of 15 questions. All questions were related to the modeled domain. Task 2 included both multiple choice and short answer questions. We asked them to record the start and end times for solving this task to be able to calculate their efficiency. We instructed them not to switch between the tasks once they started task 2 as the time should be solely allocated for answering the domain questions. However, we allowed the participants to go back to task 1 after finishing task 2 questions if they felt the need to add or fix something in their object diagrams. From the beginning and throughout the experiment, we encouraged the model receptors to ask any questions they had. We recorded all questions but we only answered questions related to tooling or understanding of requirements.

We used MagicDraw [2] for UML modeling. We introduced the tool during the first 15 minutes of the session and provided training materials on using the tool for opening the provided diagrams and creating object diagrams.

## 3.6 Operation

The operation of the experiment covered the period from November 2012 till July 2013 and was divided into two phases: the pilot study and the experiment. We first conducted a pilot study with three participants, which allowed us to test and improve the experimental design. We also conducted a test session to simulate the actual experiment to verify the clarity of the study materials and the time needed for completing the tasks. We did not include any of these data points in the results. In total, we conducted 10 experimental sessions, each having between two and three model receptors, who were randomly assigned to either the control or EDM groups.

# 4. DATA COLLECTION

## 4.1 Model Abstractions and Examples

We gathered data about loyalty programs in general and two specific programs: Club Sobeys and Shoppers Optimum. We chose these programs due to their publicly available data on their stores' websites. Each program generally has a loyalty card, sometimes called a points card, associated with it. This card identifies the card holder as a loyalty program memeber. Members earn points for every paid transaction, when they swipe their card upon checkout. Members can then redeem their points via one of the redemption channels provided in the store. One example of points redemption is instant savings off a bill.

The researcher playing the role of a SME studied the actual loyalty programs, and then the other researcher playing the role of the model creator elicited all the necessary knowledge through conversations with the SME. The model creator created a single UML class diagram that satisfied the requirements for both loyalty programs, and validated it with the SME. The class diagram was composed of 23 classes, 34 associations, and 8 constraints. The constraints were written in natural language because we expected that most model receptors would not be familiar with the object constraint language (OCL) [38]. An extract from the class diagram is shown in Fig. 3(a). The full class diagram is available with the rest of experimental materials in [4].

The SME prepared six partial examples in the form of partial UML object diagrams. Each partial example clarified one or two main concepts derived from the model abstractions. We used a variety of examples: four valid and two invalid examples. Example 6 shown in Fig. 3(b), represents an invalid example that violates a constraint; a member cannot earn and redeem points for the same bill. We see that `John` bought an `Omega3-bottle` that didn't have an offer associated with it, and hence was added directly to his bill. He also bought `VitaminC`, which was offered at a reduced price: $1 off. John knew about this offer through the `InStore-Flyer`. The bill includes the reduced price for `VitaminC` in its `dollarAmountBeforeTax` attribute value. Since there are no `BonusPoint`s associated with any offer picked by `John`, the bill only accumulated regular `BasePoint`s. We chose 10 points for every $1 as our `BaseMechanic`. This part of the example shows how a member can accumulate points. However, the rest of the example shows points redemption as well that would make `John` only pay $6.95 instead of $16.95. This represents a constraint violation: if `RedemptionMechanic` is applied, it has to be the only mechanic applied to the bill.

## 4.2 Participants' Information

We identified three factors that could have an influence on the model receptors' performance in our experiment: their background, UML experience level, and domain knowledge. The *background* refers to the working context of the model receptor, which we divided into two main categories: *industry* and *academic*. For this experiment, our model receptors were students only. However, many of them had industry experience. The *domain knowledge* refers to the extent of their familiarity with rewards loyalty programs.

Before the experiment, we collected both personal (e.g., name, email address) and professional data (e.g., department, graduate or undergraduate student, UML experience, rewards loyalty programs experience) about the model receptors using an online questionnaire. We measured and controlled the model receptors' experience levels with UML class and object diagrams as well as the problem domain through the following procedure. First, we sent them recruitment emails that mentioned our interest in recruiting students who have UML class and object diagrams experience to take part in a structural modeling experiment, and asked interested recipients to complete a background questionnaire. The questionnaire included multiple choice questions asking students to rate their expertise with structural modeling in general and with UML class and object diagrams, and their familiarity with the application domain. One question asked the students to choose the source of their experience with UML: academic, industrial, or both.

On a 5-point Likert Scale [24] ranging from 1 (no experience) to 5 (expert), we selected students who rated themselves on average: 2 or higher for structural modeling experience, 3 or higher for UML class diagram experience, 3 or higher for UML object diagram experience, and 1 for their familiarity with the application domain. Second, we sent the selected students a UML assessment exercise. The exercise aimed at making sure that the students rated their experi-

ence, with respect to UML class and object diagrams, correctly. The exercise included 11 multiple choice questions, and one diagram construction question. We selected the students who scored 75 percent or higher in the assessment exercise to be model receptors in our experiment. Among the 26 selected students, only two scored below 80 percent.

### 4.3 Timing Data

To time the model receptors accurately, the experimenter started each session for all students at the same time so that the total time allocated for all students is exactly three hours. To time the second task of the experiment, the model receptors were asked to inform the experimenter when they wanted to start solving task 2. The experimenter recorded the start and end times for the second task, and ensured that there was no switching between the tasks of the experiment.

### 4.4 Evaluation of Task Solutions

At the end of each session, the experimenter collected a soft copy of the students' object diagrams, and a hard copy of their answers for task 2. Before grading their solutions, we created a list of classes and associations that needed to be instantiated in an object diagram, a set of correct answers for task 2, and the marking scheme for each task.

We employed a blind marking technique to rule out any sort of bias; when grading the task solutions the experimenter didn't know the name of the participant nor which group he or she belonged to. After marking all solutions, the experimenter created code names for some students and created a mapping between the code names and the students groups. The code names were for sample students' solutions provided to the second and third authors in addition to the employed marking scheme, which allowed them to perform blind verification of the grades. The authors discussed the differences and agreed on the final grading.

### 4.5 Participants' Feedback

Each experiment session ended with a debriefing part, in which the model receptors assessed the level of difficulty for each task, identified specific parts of the class diagram that were hard to understand, assessed the confidence level in their answers for each task, provided feedback regarding the overall time pressure of the experiment, and, optionally, provided any comments they might have that could potentially help us improve the experiment. The model receptors who were part of the EDM group, in addition to the previous items, also answered questions related to whether it would have been hard to understand the model abstractions without examples, specified which examples (if any) were of help, commented on whether having a variety of valid and invalid examples was better than having valid examples only, commented on whether having partial examples was better than having a single complete, but big, example object diagram, and finally provided their comments regarding the number of examples used, the concept(s) covered by each example, and the size of each provided example.

## 5. DATA ANALYSIS

### 5.1 Subject Analysis

We conducted the experiment with 26 participants in several runs. We excluded one data point from the control participants as the participant couldn't finish the experiment within the allocated time. Hence, we ended with 13 data points for the EDM treatment and 12 data points for the control treatment. Moreover, the random assignment of model receptors to treatments led to a balanced distribution of graduate and undergraduate students, in addition to a balanced distribution of students' expertise among treatments.

### 5.2 Preliminary Data Analysis

From the participants' solutions for task 1, we observed that most mistakes and missing objects from the control group object diagrams were related to the classes `RedemptionMechanic`, `RedemptionChannel`, `BonusMechanic`, and `PartnerOffer`. All participants from the control group scored zero for the `RedemptionMechanic` class, and 9 out of 12 participants scored zero for incorrect or missing instances of the other two classes. On the other hand, all EDM participants got the full mark for the classes `PartnerOffer` and `RedemptionMechanic`, and 11 out of 13 participants got the full mark for the class `BonusMechanic`. The inability to correctly instantiate the class `RedemptionMechanic` by the control group indicates the usefulness of explicit examples over any other form of extra information. In the experimental materials, we provided all participants with a textual description of possible ways by which members can redeem their points: *"Members can redeem their points for instant savings off a bill or have them automatically converted to one of Sobey's partners."*, *"Possible ways by which a member can redeem points previously added to their account include: inside the store to reduce overall grocery costs"*, or through a partner such as Aeroplan: *"Aeroplan members collect miles via credit cards or via a points conversion system with one of their partners."*

We observed that all participants from both treatments answered *Q9* correctly [4]. *Q9* was about the difference between *FixedPriceOff* and *FixedPercentOff* mechanics, which are likely to be familiar to the participants based on previous shopping experience. This supports our earlier discussion from Section 2 that people who are already familiar with the domain might not need examples to improve their model comprehension compared to novices to the application domain. There are three questions that all EDM participants solved correctly, while less than half of the participants in the control group were able to solve: *Q5, Q8, Q10*. We observed the highest discrepancy between the scores of the control group for *Q15*, with a minimum score of 0 out of 5, and a maximum score of 4. This question tested the participants' ability to perform impact analysis. If they had to change the value of the *BaseMechanic* instance, then which other instances would have to be changed accordingly. We didn't observe this discrepancy for the EDM group answers for this or any other question in task 2.

## 6. RESULTS

### 6.1 Quantitative Analysis

Given the design of our experiment; a between-subjects, unbalanced design (13 students for EDM versus 12 students for the control group) with one independent variable, the suitable parametric test for hypothesis testing is the independent samples students' t-test [20]. We ensured that our data met the test assumptions through the following:

- *Independence of observations:* We met this assumption through our choice of a between-subjects design.
- *Normality of the populations across the dependent variables:* We tested the normality of all dependent variables using the normal quantile-comparison plots [20]. The assumption was met for all cases.
- *Equal variances of the populations across the dependent variables:* We tested our data for equal variances using the Levene's test [28]. The assumption was not met for diagram completeness. We used the Welch's t-test [5] for that case, which is an adaptation of the student's t-test intended for use when the two populations have unequal variances.

We performed all tests using the R statistical package [3]. We chose a significance level of 0.05 which corresponds to to a 95% confidence interval. The statistics related to *diagram completeness*, *diagram mistakes*, *task 2 completeness*, and *task 2 efficiency* are presented in Table 2.

**Diagram Completeness.** We use this variable to indicate how complete the object diagrams created by the model receptors are with respect to the provided class diagram and a set of requirements. Creating an object diagram with at least one instance for every class, all attribute values, and all links was a requirement for all participants. *We measure completeness as a percentage of the participant's score for correctly instantiated objects and links over the total reference score for all required objects and links.* By "correctly" we mean an instantiation that satisfies all constraints and domain requirements.

We calculate the total score for completeness as follows: 1 mark for each correctly instantiated object, 1 mark for each correctly included attribute value, and 1 mark for each correctly assigned link in the object diagram. `PromotionVehicle` is the only object assigned 2 marks as the provided requirements asked the participants to include *different ways for promoting in store and partner offers.* This means at least 2 different instances of `PromotionVehicle` are needed. There is 1 extra mark to satisfy the requirement that the object diagram must include at least one `OwnerProduct` instance that isn't associated with offers. Also, `firstName` and `lastName` attributes are only marked once whether the participant included them in the `Member` object or in the `Account` object. We excluded any redundancy from our calculations. For example, if a participant instantiated the same object twice, we only mark one correct instance. This allows us to have a point of comparison with other participants' diagrams. The maximum score is 69, corresponding to a 100% for diagram completeness.

**Result:** There was a significant effect of examples on diagram completeness, with a *p-value*$<$0.0002, indicating that the mean completeness score for the EDM group was significantly higher than the completeness score for the control group. This effect is illustrated in Fig. 1.

**Diagram Mistakes.** We use this variable to indicate the *number of mistakes per object diagram.* By mistake, we mean wrong interpretation and/or usage of classes, associations, in addition to unsatisfied constraints. For example, one participant used a product manufacturer such as *Kraft Foods* to be the `ProgramOwner` instead of *ClubSobeys.* Another mistake would be applying accumulation and redemption techniques for the same instance of `Bill`, which violates a constraint in the class diagram. The difference between the two variables *Diagram Completeness* and the *Diagram Mistakes* is that the first takes into account missing classes, attributes, and/or links. For example, if a participant did not include the value of an attribute, it is not considered a mistake but decreases their score for diagram completeness.

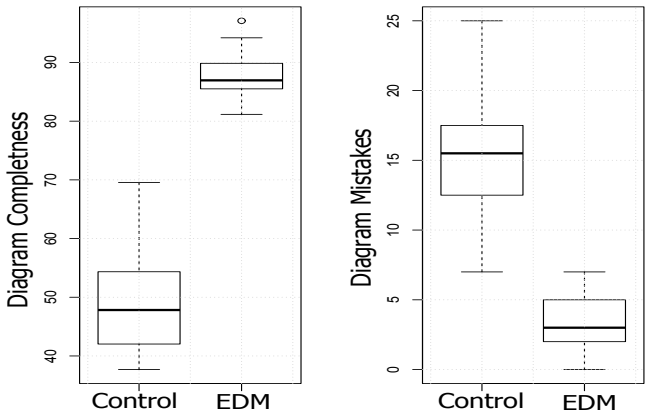



**Figure 1: Diagram completeness and mistakes.**

**Result:** As illustrated in Fig. 1, with a *p-value*$<$0.0053, the data indicates that there was a significant effect of examples in reducing the number of mistakes in the participants' diagrams who belonged to the EDM group compared to those of the control group.

**Task 2 Completeness.** This variable measures each *participant's score for answering task 2 questions correctly.* The questions given in task 2 require both model and domain understanding. Having them as the second task is important so that the participants would have already spent time understanding the model abstractions and the domain to create the object diagram in the first task. There is only one correct choice for each multiple choice question and one correct answer for each short answer question except for one question which had two possible correct answers, as seen in [4]. The score for task 2 is calculated as follows: 1 mark for each multiple choice question and 1 mark for each concept that needs to be included in the participant's answer for the short answer questions. For example, the answer for the question *"What are the type(s) of points that are collected by the bill?"*, is marked out of 2: 1 mark for *BasePoints* and 1 mark for *BonusPoints.* We gave partial credit for partially correct answers. The maximum score of correctness is 24.

**Result:** We observed an expected significant effect of examples on the participants' solutions for task 2, with a *p-value*$<$0.0012, indicating that the average task 2 completeness score for the EDM group was significantly higher than the one for the control group as shown in Figure 2.

**Efficiency.** In our analysis, we use efficiency of the model receptors in solving the questions in task 2 as a dependent variable. *We measure efficiency as the ratio between task 2 completeness score and the time spent in minutes completing the task.* The boxplot in Fig. 2 shows the efficiency scores for the EDM group compared to the control group.

**Result:** With a *p-value*=0.00082, the data shows that there was a significant effect on improving the efficiency scores for the EDM group compared to the control group.

## 6.2 Qualitative Analysis

**Table 2: Statistics related to diagram completeness and mistakes, task 2 completeness and efficiency.**

| Dep. var. | Diagram Completeness | | Diagram Mistakes | | Task 2 Completeness | | Efficiency | |
|---|---|---|---|---|---|---|---|---|
| Treatment | EDM | Control | EDM | Control | EDM | Control | EDM | Control |
| **mean** | 88.1% | 49.2% | 3.5 | 15.1 | 21.2 | 13.6 | 1.2 | 0.7 |
| **min** | 81.2% | 37.7% | 0.0 | 7.0 | 17.0 | 9.0 | 0.7 | 0.2 |
| **max** | 97.1% | 69.6% | 7.0 | 25.0 | 24.0 | 19.0 | 2.3 | 1.1 |
| **median** | 87.0% | 47.8% | 3.0 | 15.5 | 21.0 | 14.0 | 1.1 | 0.6 |
| **stdev** | 4.3% | 9.3% | 2.0 | 4.7 | 2.0 | 3.2 | 0.5 | 0.3 |

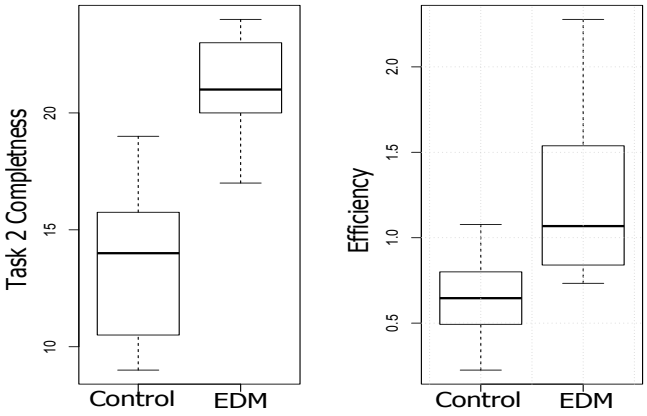



**Figure 2: Task 2 completeness and efficiency.**

We present our qualitative analysis in terms of the participant's feedback provided at the end of each experimental session. First, we asked all participants to rate the difficulty level of their experience in creating the object diagram and in solving task 2 questions. On a 5-point Likert scale ranging from 1 (very easy) to 5 (very difficult), participants from the EDM group rated their experience on average 2.46 for difficulty in creating the object diagram and 2.08 for difficulty in answering task 2 questions. Participants from the control group, on the other hand, rated their experience on average 3.83 for difficulty in creating the object diagram and 3.25 for difficulty in answering task 2 questions, which indicates a lower perceived difficulty by the model receptors to comprehend the model abstractions and solve the comprehension tasks when they had examples. We verified this when we asked the EDM participants: *"Do you think it would have been hard to understand the class diagram without examples?"*. All participants answered *"Yes"*. The difference in experienced difficulty by participants was apparent in the number of questions about the domain raised by the EDM versus the control group. On average, each session had 10 domain questions raised by the control group compared to one domain question for the EDM group (reduction of 90%).

Second, we asked the participants to indicate the classes and/or associations that were most difficult to comprehend. Most of the control group answers included the following classes: `BonusMechanic`, `RedemptionMechanic`, `PartnerOffer`, and `Bill`. As for the most difficult associations, these were the associations involving these classes. Some of the reasons the control group found these tasks and associations difficult were: *"I couldn't understand how the attributes propagate through the classes. Where shall I start with points calculation?"*, *"When do I use each of the four associations for the Points class? It's confusing without seeing them in context."*, *"I started with a few classes to understand them, but whenever I found a constraint it made me change the way I visualized how the classes are linked together and hence a total shift in my understanding."*, and finally *"the concepts for these classes unlike Member or Account classes don't hold any previous meaning and hence were difficult to get their idea. Although I understood the meaning of Bill and saw the formulas for points calculation, I couldn't form a picture of how the classes are combined to form a system for accumulating points."*

The EDM group mentioned that the same classes and associations were hard to comprehend from the abstractions only, but the examples helped a lot. We asked the EDM participants to list the examples they found most useful and helpful. The majority mentioned examples 2, 4, 5, and 6 [4]. These examples covered all classes and associations that the control group participants found hard to understand. The participants found these examples most useful for the following reasons: *"example 2 shows separately how a regular product is treated compared to a one that has an offer"*, *"example 2 shows how to apply bonus mechanic when you are using the multiplier attribute. It also shows that the pointsMultiplier and fixedAmount attributes are mutually exclusive"*, *"example 4 shows how the attributes from different classes relate and the sequence for accumulating points. For example, you have to finish in-store points before you apply partner points"*, *"example 4 shows that the account points balance gets updates with which value in the Bill class. For example, it doesn't get the base or bonus points directly"*, and *"I couldn't have figured out any classes from the redemption process except through this example [example 5]"*.

Twelve out of 13 EDM participants, when asked whether having a variety of valid and invalid examples was more helpful than having valid examples only, replied that they found the invalid example 6 helpful in understanding the constraint about how one can not apply accumulation and redemption for the same bill. The only participant who did not find the invalid examples useful answered incorrectly the following question in task 2 related to the constraint violation illustrated by example 6: *"Can a customer earn and redeem points for the same bill?"*. We also asked the EDM participants if having small object diagrams representing partial examples was more useful than having one complete object diagram. They all preferred the small examples, but one participant preferred if he would have had the complete object diagram in addition to the small partial examples as it would have helped him integrate all the concepts together.

Finally, we asked the participants to rate their confidence level in the answers they provided for both experimental tasks. On a 5-point Likert scale ranging from 1 (unsure) to 5 (very sure), the EDM group participants rated their confidence level on average 4.08 for their object diagram answers and 4.15 for their task 2 answers. The control group participants rated their confidence level on average

2.33 for their object diagram answers and 3.00 for their task 2 answers. These results indicate that model receptors who see model abstractions together with examples experience a deeper level of understanding which makes them more confident in their answers.

During feedback elicitation, the participants from both groups commented on the use of UML or any graphical modeling notation in general. They mentioned that part of the experiment difficulty was not knowing where to start reading the model abstractions. The class diagram was of significant size with lots of concepts, classes, associations, and constraints to digest. The EDM group mentioned that partial examples were very useful in that situation since partial examples helped them focus on a small subset of the class diagram at a time. One participant commented that the order by which the examples were presented was very important and helped in smoothing the comprehension process where first they got introduced to the concept of offer which is the basic class for accumulation, then a regular product in addition to an offer to emphasize the difference in points calculation between a regular product and a product having an offer, then a constraint about in-store points accumulation, then partner offers, followed by redemption as he started reading the class diagram from left to right, and finally the relation between accumulation and redemption. We asked the EDM participants if they preferred more examples, but they all agreed that the provided examples were enough to understand the class diagram and perform the comprehension tasks. This suggests that one positive example per concept is enough unless there is more than one idea associated with that concept. For example, we had 3 positive examples for accumulation, but only one for redemption. Also, one wouldn't need an invalid example for every constraint, but only when the constraints are related to more than one concept at a time or the ones which express corner cases.

## 6.3 Discussion

In addition to the quantitative and qualitative analysis, we performed a detailed task analysis, which provided a number of insights on our experimental design, and the type of tasks that our approach supports best. Most participants from the EDM group agreed that the number and choice of examples were excellent. They had no comments or issues related to a concept that was not conveyed in the examples, nor that the number of examples was too small or too big. The only participant who preferred valid examples only did solve the question related to the invalid example (constraint) incorrectly. All participants agreed that partial examples were better than a single complete object diagram since this helped them focus on a small subset of the model at a time. The examples benefited the model receptors the most when there were several associations related to a single class, propagation of attributes through classes, unfamiliar domain concepts, and understanding the system's constraints.

Through our detailed task analysis, we wanted to see whether the effect of examples on task 2 completeness was independent from the fact that all model receptors did an instantiation for task 1 first. Nine out of 13 participants for the EDM group chose the *fixedAmount* attribute for the *BonusMechanic* class when instantiating their object diagram. They were still able to solve *Q10* and *Q12* that asked about the optional *pointsMultiplier* attribute correctly, despite not having used that attribute in instantiation. Also, although some participants had mistakes or missing classes for *Bill* or *FixedPerecentOff* mechanics, they were still able to solve the related questions in task 2 correctly. This shows that when using examples, the participants were able to solve task 2 correctly despite having mistakes in the instantiation task, which suggests that performing instantiation first might not be necessary for the participants ability to answer task 2 questions correctly.

Finally, the experiment tested EDM only from cause and effect viewpoints only without considering the cost. Although the presence of examples led to a better performance in terms of diagram completeness, diagram mistakes, task 2 completeness, and task 2 efficiency, we didn't take into consideration the cost of creating and maintaining the examples in terms of time, effort, or availability of tool support. We need to investigate this matter in an industrial context.

# 7. THREATS TO VALIDITY

This section discusses the threats to internal, external, construct, and conclusion validity according to [20, 40].

## 7.1 Internal Validity

Threats to internal validity refer to uncontrolled factors in the environment that may influence the effects of the treatments on the variables.

**Participants.** We ensured that the participants were familiar with the class diagram constructs used in the experiment using a UML assessment exercise in order to reduce the threat that model receptors may not have been competent enough in the modeling notation. Each participant applied one treatment only to avoid the human learning effect. We used randomization to assign participants to treatments in order to mitigate the effect that the participants' experience may not have been fairly distributed among treatments. We prevented communication between the participants during the session in order to avoid one participant's answers affecting the other. We did not inform the participants of our experimental hypotheses nor what measures we were looking for to avoid their expectations from biasing the results. To compensate for the effect of the participants' knowledge of the application domain on the dependent variables, we selected the participants who evaluated themselves as having no experience with rewards loyalty programs.

**Tasks.** Fatigue during completion of task 2 questions is a possible threat to validity. The number of wrong answers for both groups is almost the same for questions at the beginning and questions at the end. Therefore, there is no evidence of any decrease in performance. The choice of tasks may have been biased to the advantage of EDM. We alleviate this threat by selecting tasks that target what developers/BAs do in real life when they don't understand the system in the absence of a SME. For example, questions like *"what are the classes which represent different means of accumulating points?"* are considered with the concept location task. If a novice to the application domain needs to understand how a member in a loyalty program can earn points, they will have to determine the different classes associated with points accumulation. Another targeted activity was impact analysis. By impact analysis we mean determining the impact of a change on a system. This activity requires full understanding of the system. Question 15 [4] targeted this activity. We gave all participants the same

amount of time to finish both tasks of the experiment for fair comparison. This led to the exclusion of one data point from the control group when the participant exceeded the allocated time for the experiment.

**Session Differences.** There were several runs for the experiment to accommodate the availability of the participants and the differences between them may have influenced the results. To mitigate this threat, we had participants from both EDM and control treatments in each session. The pilot study also helped us obtain a stable and reliable setup.

## 7.2 External Validity

External validity threats are concerned with whether the results can be generalized outside the experimental setting.

**Participants.** To mitigate the threat of the representativeness of the participants, we could only address their experience level with UML. However, we were not able to include participants from industry.

**Problem Domain.** The representativeness of our domain is another threat. We chose to perform the experiment with a domain that was inspired from a similar domain used by one of our industrial partners. The process by which the domain was modeled was also inspired from the industry, where the modeler usually has elicitation sessions with the SME to understand the domain.

**Tasks.** The choice of tasks may not precisely reflect what practitioners do with the models, but they certainly reflect the degree to which participants comprehend the models with and without examples.

**Experimenter Effect.** The experimenter is also one of the authors of this paper. This could have influenced the experiment. One instance of this threat is grading the task solutions correctly. To mitigate this threat, we created a marking scheme before grading the participants' solutions and performed double marking. Moreover, the experimenters graded the solutions blindly.

## 7.3 Construct Validity

Construct validity is the degree to which the dependent and independent variables represent the cause and effect concepts that should be measured in the experiment.

**Participants.** In the pilot study, we relied only on the participants' own rating of their expertise in UML class and object modeling. However, in order to ensure that we accurately measured the skill level of the participants, we required them to solve a UML assessment exercise.

**Tasks.** We avoided mono-operation bias by having more than one subject perform each treatment, and more than one measure of comprehension. Moreover, we limited the number of multiple choice questions in task 2 to five questions, while the rest were short answer questions. This helped imitating a real-life context whereby the stakeholders are not guided by a set of predefined interpretations.

**Experimental Materials.** We avoided the threat that the EDM participants would use the partial examples as templates for their object diagrams instead of using them to improve their model understanding through the following: (1) the combination of partial examples could not be integrated into a full object diagram as we used different instantiations for the same class (i.e. different alternatives for the same class) in different examples, (2) we used invalid examples, which could not be used as templates as they would lead to incorrect system behaviour, and finally (3) the participants would still need to understand the examples to solve task 2 questions. Our results show that the EDM participants' answers to task 2 were of significant improvement over the answers provided by the control participants.

## 7.4 Conclusion Validity

Conclusion validity is the extent to which correct conclusions are drawn about the relationship between the treatments and the outcomes. The treatments assume novices to the application domain who are familiar with the modeling notation, so we provided homogeneous participants' groups with respect to their background and expertise. The statistical tests used to draw our conclusions have assumptions that needed to be satisfied before they could be applied, so we verified that the assumptions of the tests before we used them in our analysis. Accurate measurements of treatment outcomes are necessary to reflect the effects, so we used a marking scheme, blind and double marking techniques.

# 8. RELATED WORK

We started the experiment by conducting a survey of research work dealing with experimental validation of software engineering, model comprehension approaches, and UML modeling studies related to class and object diagrams. There is a rich body of research on empirical evaluation of model comprehension techniques especially with respect to UML class diagrams. The experiment presented in [41] tests the effect of different layouts strategies on UML model comprehension. The results indicate improved accuracy in solving comprehension tasks when having clustered layouts. Similarly, the experiment presented in [26] focuses on how the level of detail in UML models impacts model comprehension. The results show an improved effect of the level of detail on the comprehension of UML models.

Although there exist many studies on model comprehension, there are only two studies, according to our knowledge, that assess the effect of having object diagrams together with class diagrams on model comprehension. The first study [35], used four comprehension tasks to determine the impact of object diagrams on model comprehension. Each task was composed of multiple choice questions for a class diagram different than the class diagrams used in the other tasks. The study used only multiple choice questions, which increases the threat of correct results due to participants guessing the answers. The participants were allowed to communicate during the study and hence one participant's answers might have affected the other. Also, each class diagram was composed of only 4 classes which is significantly simpler than any model representing real-world systems. The results show an improved effect on comprehension only for two tasks out of the four at 85% confidence interval. The second study [34] is a replication of the first one, but with minor modifications. The researchers changed the questions to be open ended and added a dependent variable; the amount of time it took each participant to solve each task. However, this dependent variable doesn't take into account that a participant can take longer time to answer the questions, and hence achieve a higher score. The authors still allowed participants to communicate, and the four class diagrams were still the same. The results show an improved effect of object diagrams on comprehension at 95% confidence interval.

In this work, we mitigated the above threats to validity by

selecting participants who were novices to the domain to prevent guessing; presenting the model receptors a significantly more complex class diagram; preventing any communication among the participants; including an instantiation task before questions; including both multiple choice questions and open-ended questions; measuring multiple and diverse dependent variables, both quantitatively and qualitatively, to present a fuller picture of the effects.

# 9. CONCLUSIONS

We presented a controlled experiment that aimed at evaluating the effects of using explicit examples on model and domain comprehension in the context of our proposed structural modeling approach, EDM. We used the domain of rewards loyalty programs. We represented the abstractions part of the model as a UML class diagram, and our examples as six partial UML object diagrams. Our participants were 26 graduate and undergraduate students from the University of Waterloo.

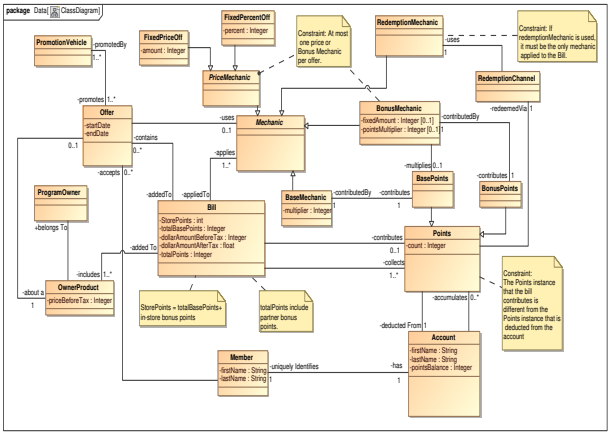
The main result of our experiment is that the EDM approach leads to improvement in all measured variables: diagram completeness (+39%), diagram mistakes (-80%), task 2 completeness (+30%) and efficiency (+71%). If on average a participant in the EDM group has 3 mistakes in his diagram, while a participant in the control group has 15, then the EDM participant has 80% fewer mistakes. For task 2 completeness, we converted the average scores for both EDM and control as a percentage of the total score (25 marks), and then calculated the difference as a percent. This result is statistically significant, which allows us to reject the first four null hypotheses and accept the corresponding four alternative hypotheses. We conclude that this improvement in model and consequently domain comprehension is due to augmenting the model abstractions with explicit examples.

We also performed a qualitative analysis for the last three null hypotheses. The EDM participants experienced a reduction in the perceived difficulty for creating the object diagram (-27%), and a reduction in the perceived difficulty for solving task 2 questions (-23%). They also experienced an increase in confidence in their created diagrams (+35%), and an increase in confidence in their task 2 solutions (+23%), while asking fewer domain questions (-90%), as compared to the participants in the control group. Although we can not reject the fifth and sixth null hypotheses based on qualitative results, these results provide insight on the impact of having examples on the model receptors' experience in understanding the model abstractions and the domain when they are provided with examples.

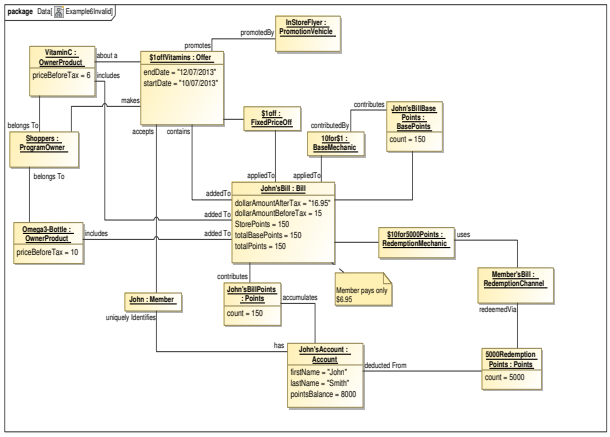
Our analysis suggests that examples are needed the most when the concepts of the model abstractions are not familiar to the model receptor. One positive example per concept is generally enough unless there is more than one idea formulating that concept such as with points accumulation or when several associations are associated with a class, but are applied under different conditions. Negative examples are only needed for corner case constraints that are not readily understandable. Partial examples help the model receptors focus on a small subset of a bigger model at a time which improves domain comprehension. However, for future work we might need to consider having partial examples as views on more complete examples as this might help with integrating the different concepts for a better abstracted mental model. The future work should also evaluate the effects of using a variety of valid and invalid examples as compared to having valid examples only, and verify the usefulness and the cost/benefit ratio of applying EDM in an industrial context. Our results also provide support for the importance of modeling language design. We propose that structural modeling languages should not only have explicit notations for examples or instances as an integral part of the model, but also support partial examples. In other line of work, we designed a lightweight structural modeling language called Clafer [1, 9, 7, 8] and showed how it could be used for EDM [6].

Finally, we designed our experiment with replication in mind. We published supplemental meterials [4] consisting of the UML class diagram, the set of six partial UML example object diagrams, the experimental materials for both the control and EDM treatments, the de-briefing questionnaire, and the grading scheme for both experimental tasks. This allows the readers to better evaluate our experimental design and the presented results. It also allows other researchers to replicate the experiment, or benefit from the design as the base for their own.

# APPENDIX




(a) Class diagram extract.




(b) Partial object diagram representing an invalid partial example: accumulation and redemption can not be applied for the same bill.

**Figure 3: UML model and example extracts.**

# A. REFERENCES


[1] Clafer Homepage. `http://clafer.org`. Accessed: 2014-02-15.

[2] Magicdraw. `http://www.nomagic.com/products/magicdraw.html`. Accessed: 2014-02-15.

[3] The R project for statitical computing. `http://www.r-project.org/`. Accessed: 2014-02-15.

[4] Supplemental materials for the controlled experiement presented in Zayan et al., *Effects of Using Examples on Structural Model Comprehension, 2013*. `http://gsd.uwaterloo.ca/node/541`. Accessed: 2014-02-15.

[5] Welch's t-test. `http://statistics.berkeley.edu/computing/r-t-tests`. Accessed: 2014-02-15.

[6] M. Antkiewicz, K. Bąk, D. Zayan, K. Czarnecki, A. Wąsowski, and Z. Diskin. Example-driven modeling using Clafer. In *First International Workshop on Model-driven Engineering By Example*, 2013.

[7] K. Bąk. *Modeling and Analysis of Software Product Line Variability in Clafer*. PhD thesis, University of Waterloo, 11/2013 2013.

[8] K. Bąk, Z. Diskin, M. Antkiewicz, K. Czarnecki, and A. Wąsowski. Partial instances via subclassing. In *6th International Conference on Software Language Engineering*, 2013.

[9] K. Bąk, K. Czarnecki, and A. Wąsowski. Feature and meta-models in clafer: Mixed, specialized and coupled. In *International Conference on Software Language Engineering*, pages 291–301, 2010.

[10] K. Bąk, D. Zayan, K. Czarnecki, M. Antkiewicz, Z. Diskin, A. Wąsowski, and D. Rayside. Example-driven modeling. Model = Abstractions + Examples. In *New Ideas and Emerging Results (NIER) track of ICSE'13*, 2013.

[11] N. Cerpa and J. M. Verner. Why did your project fail? *Commun. ACM*, 52(12):130–134, Dec. 2009.

[12] M. T. Chi, P. J. Feltovich, and R. Glaser. Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5(2):121 – 152, 1981.

[13] J. Dick and A. Faivre. Automating the generation and sequencing of test cases from model-based specifications. In J. Woodcock and P. Larsen, editors, *FME '93: Industrial-Strength Formal Methods*, volume 670 of *Lecture Notes in Computer Science*, pages 268–284. Springer Berlin Heidelberg, 1993.

[14] B. Dobing and J. Parsons. How UML is used. *Commun. ACM*, 49(5):109–113, May 2006.

[15] M. L. Gick and K. J. Holyoak. Schema induction and analogical transfer. *Cognitive Psychology*, 15(1):1–38, 1983.

[16] M. L. Gick and K. Paterson. Do contrasting examples facilitate schema acquisition and analogical transfer? *Canadian Journal of Psychology*, 46:539–550, 1992. 4.

[17] R. L. Goldstone and J. Y. Son. The transfer of scientific principles using concrete and idealized simulations. *The Journal of The Learning Sciences*, 14(1), 2005.

[18] J. Hutchinson, J. Whittle, M. Rouncefield, and S. Kristoffersen. Empirical assessment of MDE in industry. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 471–480, New York, NY, USA, 2011. ACM.

[19] D. Janzen and H. Saiedian. Test-driven development concepts, taxonomy, and future direction. *Computer*, 38(9):43–50, 2005.

[20] N. Juristo and A. M. Moreno. *Basics of Software Engineering Experimentation*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[21] S. Kalyuga. Knowledge elaboration: A cognitive load perspective. *Learning and Instruction*, 19(5):402 – 410, 2009. Cognitive load in interactive knowledge construction.

[22] B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. El Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *Software Engineering, IEEE Transactions on*, 28(8):721–734, 2002.

[23] S. Lauesen and O. Vinter. Preventing requirement defects: An experiment in process improvement. *Requirements Engineering*, 6(1):37–50, 2001.

[24] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 140:1–55, 1932.

[25] D. North. Introducing BDD. *Better Software*, 12, 2006.

[26] A. Nugroho. Level of detail in UML models and its impact on model comprehension: A controlled experiment. *Inf. Softw. Technol.*, 51(12):1670–1685, Dec. 2009.

[27] J. Offutt and A. Abdurazik. Generating tests from UML specifications. In R. France and B. Rumpe, editors, *UML'99—The Unified Modeling Language*, volume 1723 of *Lecture Notes in Computer Science*, pages 416–429. 1999.

[28] I. Olkin. *Contributions to probability and statistics: essays in honor of Harold Hotelling*. Stanford University Press, 1960.

[29] S. S. Park. Communicating Domain Knowledge through Example-Driven Story Testing, 2011.

[30] M. Petre. UML in practice. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 722–731, 2013.

[31] H. C. Purchase, L. Colpoys, M. McGill, D. Carrington, and C. Britton. UML class diagram syntax: an empirical study of comprehension. In *Proceedings of the 2001 Asia-Pacific symposium on Information visualisation - Volume 9*, APVis'01, pages 113–120, Darlinghurst, Australia, Australia, 2001. Australian Computer Society, Inc.

[32] F. Ricca, M. Di Penta, M. Torchiano, P. Tonella, and M. Ceccato. The role of experience and ability in comprehension tasks supported by UML stereotypes. In *29th International Conference on Software Engineering*, pages 375–384, 2007.

[33] R. Salay, M. Famelis, and M. Chechik. Language independent refinement using partial modeling. In *FASE*, volume 7212 of *Lecture Notes in Computer Science*, pages 224–239, 2012.

[34] G. Scanniello, F. Ricca, and M. Torchiano. On the effectiveness of the UML object diagrams: A replicated experiment. In *Evaluation Assessment in Software Engineering (EASE 2011), 15th Annual*

*Conference on*, pages 76–85, 2011.

[35] M. Torchiano. Empirical assessment of UML static object diagrams. In *Program Comprehension, 2004. Proceedings. 12th IEEE International Workshop on*, pages 226–230, 2004.

[36] T. van Gog, L. Kester, and F. Paas. Effects of worked examples, example-problem, and problem-example pairs on novices' learning. *Contemporary Educational Psychology*, 36(3):212 – 218, 2011.

[37] R. van Solingen, V. Basili, G. Caldiera, and H. D. Rombach. *Goal Question Metric (GQM) Approach.* John Wiley & Sons, Inc., 2002.

[38] J. Warmer and A. Kleppe. *The Object Constraint Language: Getting Your Models Ready for MDA*.

[39] J. Whittle. What do 449 MDE practitioners think about MDE? In *EESSMod*, 2011.

[40] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering: an introduction.* Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[41] S. Yusuf, H. Kagdi, and J. I. Maletic. Assessing the comprehension of UML class diagrams via eye tracking. In *In 15th International Conference on Program Comprehension (ICPC'07)*, pages 113–122. IEEE Computer Society, 2007.