Effects of Using Examples on Structural Model Comprehension: A Controlled Experiment

by

Dina Zayan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2013

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

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 on 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 study 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, the number of examples, the types of examples, and the use of partial examples.

Acknowledgements

I would like to start by sincerely thanking my supervisor Dr. Krzysztof Czarnecki for giving me the chance to join his amazing research group. His guidance, understanding, and patience allowed me to bring out the best in myself, improve my skills and rise up to the challenge. There are not enough words that could express my gratitude and appreciation for what he did for me.

I would also like to thank Dr. Michal Antkiewicz for the countless discussions, continuous feedback, amazing advice, endless help and support through the course of my studies. He was an amazing mentor and a helpful friend.

All my love and countless thanks go to my beloved parents Omar Zayan and Mona Aziz, one of a kind husband Abdallah Elshabrawy, lovely sister Dalia Zayan and awesome little brother Mohamed Zayan. The world would not have a meaning without all of you in my life.

Sincere gratitude goes to my amazing friend Mohamed Malaika. Without you, I would not have been in such a wonderful place in my life. You inspired me to believe in myself, my choices and my heart for they know what is best for me. You taught me how to trust life and be in harmony with its tunes. Thank you for everything.

Special thanks goes to an older brother and an awesome friend Gasser Salem. Your continuous support, unconditional advice, morning messages and viber calls always encouraged me to do my best. I am grateful to have a brother like you.

I would like to thank the loveliest friends anyone can hope for Musheera Khalifa, Aya Ahraf, Sara Hesham, Marwa Sharawi, Yomna Hamza, Heba Hany, Hana Ibrahim, Roua Safwat, Ahmed Zayan, Amr Ezz, Mohammed Ibrahim, Karim Moussa, Mohamed Kerbek, Naveed khan, and Yasmine Elbahnasawy. I could not have done it without your continuous encouragement and wonderful support.

Last but not least, I would like to thank my second family Ahmed Elshabrawy, Karima Fahmy, Walaa Elshabrawy, Samah ELwakeel, Wael Elshabrawy for wishing me the best of luck in my studies. I could not have wished for better in-laws.

Acknowledgement of Contributions

Most of the material presented in this thesis was taken from existing and submitted papers. Chapter 2 is based on [7], to which my personal contribution was approximately 30%. The rest of the thesis is based on [13], to which my personal contribution was approximately 80%.

Dina Zayan

As one of the authors of [7, 13], I fully acknowledge the thesis author's statement above and give full permission to use any part of the papers we co-authored mentioned above. I also fully acknowledge that the thesis represents original research conducted by the thesis author.

Krzysztof Czarnecki

As one of the authors of [7, 13], I fully acknowledge the thesis author's statement above and give full permission to use any part of the papers we co-authored mentioned above. I also fully acknowledge that the thesis represents original research conducted by the thesis author.

Michal Antkiewicz

As one of the authors of [7, 13], I fully acknowledge the thesis author's statement above and give full permission to use any part of the papers we co-authored mentioned above. I also fully acknowledge that the thesis represents original research conducted by the thesis author.

Kacper Bak

As one of the authors of [7], I fully acknowledge the thesis author's statement above and give full permission to use any part of the papers we co-authored mentioned above. I also fully acknowledge that the thesis represents original research conducted by the thesis author.

Dedication

To my amazing parents whose unconditional love and support, have and will always guide me, to achieve my highest potential. I will forever be indebted to you for everything you did for me.

To my loving husband, the most amazing knight any girl could ever wish for. Nothing I would ever do can come close to how much support, encouragement, motivation, and love you have given me. Thank you for choosing me to be your life partner.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The process of knowledge transfer occurs in most activities of software development (e.g., analysis, design, implementation, and testing). In this thesis, we consider the transfer of domain knowledge among different project stakeholders throughout the different phases of a software project.

The main challenge for 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 possess a different set of skills [33]. Subject Matter Experts (SMEs) usually assume that business analysts (BAs) possess their same level of knowledge and are able to infer the business rules that are required to build a software system that addresses the SMEs problems. The problem is worsened when the BAs mistakenly think that they understand the domain and do not need further elicitation sessions. This problem contributes to the production of incomplete and/or incorrect requirements artifacts, which represent a major reason for the failure of software projects [23, 8].

To help with knowledge transfer, structural modeling can be used for articulating, capturing, and communicating domain knowledge [33] among the various stakeholders. However, the constructed models often contain domain abstractions only that cannot be easily validated by other stakeholders (including SMEs) who usually are not modeling experts [14]. This disadvantage to current structural modeling practices is important to address since most of the stakeholders who would actually benefit from modeling are not modeling experts.

Based on research results from cognitive psychology and software engineering, we propose that explicit examples should be used together with the model abstractions for effective domain knowledge transfer [7]. The process of collecting, communicating, and verify-

ing domain knowledge iteratively through examples is what we refer to as *example-driven modeling* (EDM) [7]. 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 (i.e., used as a communication tool between different stakeholders). In EDM, the purpose of examples is to explore and learn about the domain during all phases of software development.

1.1 Motivation

Examples are commonly used in software engineering. Test cases are examples of what the software should do. Traces are examples of behavioral models and are used to validate the models [12, 30]. Approaches such as test-driven development [19], behavior-driven development [28], story-test-driven development [33], 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 [34, 48]. For example, UML object diagrams, which represent examples of more abstract class diagrams, are rarely used in practice [34, 48].

1.2 Problem Statement

It is instrumental 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 which presentation forms are the best way to improve model, and consequently domain, comprehension.

1.3 Objectives and Contribution

The objective of this thesis is to design and execute a controlled user experiment that provides the missing empirical evidence about the effects of using examples on structural model and domain comprehension.

The main contributions provided in this thesis are the results of the controlled experiment. The results show that augmenting model abstractions with explicit examples

improved model and domain comprehension for participants who were novices to the application domain. The highest score (69.6%) for diagram completeness achieved by the participants who received the model abstractions only is lower than the lowest score (81.1%) of those who received the abstractions together with examples. 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 than the control group. EDM group participants also experienced lower perceived difficulty in 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.

1.4 Thesis Organization

For the rest of the thesis: Chapter 2 provides an overview of EDM; the proposed structural modeling technique where abstractions and examples are equally important parts of the model. Chapter 3 describes the design of our experiment, and details its operational phase. Chapter 4 describes our data collection techniques. Chapter 5 presents our preliminary data, subjects, and outlier analysis. Chapter 6 discusses the results and major findings, followed by a presentation of the threats to validity in Chapter 7. Chapter 8 discusses the related work. Concluding remarks and future work are in Chapter 9.

Chapter 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], in this chapter we present the theory from which our research questions and experimental hypotheses are derived.

Building upon results from cognitive psychology and software engineering, we previously proposed EDM [7], an approach where structural modeling is based on examples. EDM relies on the systematic use of domain examples for eliciting, modeling, communicating, and verifying complex domain knowledge among various stakeholders.

2.1 Abstract versus Concrete

Software models are general abstractions that represent many possible examples or partial system configurations that are more specific [25]. 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. EDM also distinguishes two different, but equally important modeling activities that relate abstractions to examples: abstraction inference (AI) where abstractions are synthesized from examples, and example derivation (ED) where examples are generated from abstractions.

2.2 Why Base Modeling on Examples?

1. *Examples help in improving the quality of models:*

Humans learn from examples by generalizing the information these examples present and by building mental models [25, 43]. "Example Generalization models suggest that problem solving rules are acquired while studying examples" [43]. Gick and Holyoak [15] also showed that humans learn abstractions and are able to solve problems more effectively if these problems are presented together with examples. Moreover, they showed that having multiple examples led to higher quality constructed abstractions. Schworm had similar results in [40]. By analogy, we hypothesize that explicit examples improves the quality of models by helping capture relevant domain details, and by serving as test cases for these constructed models. They should also have a positive impact on the stakeholders' (e.g., modelers, SMEs) confidence that the model is valid.

2. *Augmenting models with explicit examples improves model comprehension among various stakeholders:*

   The use of examples in current structural modeling practices, at least in the UML context, is undervalued. In practice, class diagrams are much more frequently used than object diagrams [34, 48]. This suggests that even if examples are used in the structural modeling world, they are used informally. 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 to work with models via examples [14, 18, 25]. According to Van God et. al. [44], the use of examples lowers the mental effort required to understand problems. They are easier to maintain in the working memory than abstract presentations and are more motivating for learning [25, 35].

   We hypothesize that fostering model comprehension is achieved if model abstractions are augmented with examples, since it was proved that humans experience optimum knowledge transfer when they learn abstractions together with examples [16, 17].

2.3 How to Use Examples in EDM?

1. *EDM starts with either abstractions or examples:*

   Going back and forth between abstractions and examples is unavoidable in the modeling process. A complicated, big-size model with a significant number of domain concepts, associations and constraints would be impossible to comprehend without specific examples that put it in context. At the same time, forming a mental model from a huge set of examples is very hard. One typically needs to go back and forth

between examples and abstractions for a better constructed model and better domain understanding. Abstractions and examples should evolve together. This derives the need for partial examples.

Partial examples are needed to allow expressing only the relevant information during the modeling process. They help different stakeholders focus on the relevant subset of the model that is under discussion. Partial examples can be expressed using partial instances, whereby some elements (e.g., objects, attributes, links) can be omitted 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) [4, 38]. We hypothesize that having partial examples limited to a given context is more beneficial than having full complete instances of the model.

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, 44, 9]. The difference comes from expertise which depends on the modeler's level of domain knowledge. 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.

2. *A variety of generated examples leads to better construction, comprehension, and validation of models:*

   The choice of examples is important for effective comprehension and knowledge transfer [7]. The most effective form of examples are *near-hit* and *near-miss* examples [25] as their role is to define the model's boundary. A *near-hit* example is an example which lies just inside the model's boundary (i.e., the example would be excluded if changed). On the other hand, a *near-miss* example is an example that is excluded from the model but could be included if the example is slightly changed.

   In general, positive or valid examples represent valid instantiations of the model abstractions. They show correct system behavior. On the other hand, negative or invalid examples represent incorrect system behavior. Hence, one could think of the system constraints as the abstractions of these negative examples. Constraints play an important role in reducing the model's variability and uncertainty.

   The need for a variety of examples in example-based learning techniques has been recognized in several fields such as: software testing, machine learning and grammar

inference research. In the former, a variety of different test cases provide better code coverage, which impacts the effectiveness of the employed testing process [11]. In grammar inference and machine learning, multiple examples are using for finite state automation [32] and inductive logic programming [25, 27].

Based on the current literature, we hypothesize that a variety of examples is needed to construct, validate and understand structural models.

2.4 Technical Challenges

Besides the technical improvements that are anticipated by having the consistent and systematic usage of examples in the structural modeling process, we are also facing some technical challenges: properties of examples, the order of presenting examples to stakeholders, the cost of creating and maintaining the examples, and tooling challenges.

A set of examples is complete for the AI EDM activity if we can learn everything that is needed about the model from this set of examples. A model that is abstracted from a single example is likely to be over-constrained, and an under-constrained model will likely to include undesirable examples. Coming up with not only a complete, but a minimal set of examples is important as it plays an important role in reducing the cost of model development which can be measured in terms of model completeness and development time [25]. Having a minimal set of examples is equally important in model comprehension in order to convey the needed concepts within a minimal period of time.

Covering the model with a minimal set of examples requires investigation into determining the optimum ratio of valid to invalid examples and the model boundary through generation of near-hit and near-miss examples. Since we don't know the boundary when constructing the model, it might be useful to start by generating the most contrasting examples in order to minimize the number of examples needed. Another challenge would be, when would we know that this set of examples is ‘‘just enough”?

The need for new modeling tools to support EDM is instrumental. EDM requires different tools for its different modeling activities. A BA who is unfamiliar with the domain will likely learn first from examples and would need tool support to synthesize models from examples. On the other hand, a BA who is familiar with the domain will likely start from the model abstraction and need tool support to generate examples from models (for validation and explanation). EDM also requires tools that support partial examples, which are currently handled mostly by academic tools. Furthermore, the simultaneous evolution

of models and examples (i.e., moving from abstractions to examples and vice versa) calls for tools that offer "seamless, interactive support for such co-evolution during modeling" [7].

Chapter 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 where the model consists of abstractions only. We followed the guidelines for software engineering (SE) experimentation presented by Juristo et al, Kitchenham et al, and Wohlin et al [22, 20, 49]. The goal of the experiment is summarized using the GQM template [45] (Table 3.1).

Table 3.1: Experimental goal according to GQM template.

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

3.1 Research Questions

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 over having only model abstractions?

- **RQ2:** Does using a variety of valid and invalid examples improve the model receptors' comprehension of the model abstractions over having only valid examples?

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 treatment 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 part of the control group or the EDM group.

Similar to other empirical evaluations of UML comprehension [39, 42, 20, 37, 36], the dependent variables of our experiment are *diagram completeness*, *diagram mistakes*, *study questions response completeness*, the *efficiency* in answering the questions, the participants' *perceived difficulty* in performing the comprehension tasks, and finally the participants' *perceived confidence* in their task solutions.

3.3 Hypotheses

We use several measures of comprehension as dependent variables for running the experiment. Accordingly, we formulate the following *null* hypotheses:

- $H1_0$: Augmenting model abstractions with explicit examples *does not impact* the completeness of the object diagram created by each participant.
- $H2_0$: Augmenting model abstractions with explicit examples *does not impact* the number of mistakes in a given participant's diagram.
- $H3_0$: Augmenting model abstractions with explicit examples *does not impact* the number of study questions answered correctly by each participant.
- $H4_0$: Augmenting model abstractions with explicit examples *does not impact* the participant's efficiency in solving the study questions.

- $H5_0$: Augmenting model abstractions with explicit examples *does not impact* the participants' perceived difficulty in performing the comprehension tasks.
- $H6_0$: Augmenting model abstractions with explicit examples *does not impact* the participants' perceived confidence in their comprehension tasks' solutions.
- $H7_0$: Having a variety of valid and invalid examples *does not improve* model comprehension, compared to having only valid examples.

Congruently, we formulate the following alternative hypotheses:

- $H1$: Augmenting model abstractions with explicit examples *improves* the completeness of the object diagram created by each participant.
- $H2$: Augmenting model abstractions with explicit examples *reduces* the number of mistakes in a given participant's diagram.
- $H3$: Augmenting model abstractions with explicit examples *increases* the number of study questions answered correctly by each participant.
- $H4$: Augmenting model abstractions with explicit examples *improves* the participant's efficiency in solving the study questions.
- $H5$: Augmenting model abstractions with explicit examples *reduces* the participants' perceived difficulty in performing the comprehension tasks.
- $H6$: Augmenting model abstractions with explicit examples *increases* the participants' perceived confidence in their comprehension tasks' solutions.
- $H7$: Having a variety of valid and invalid examples *improves* model comprehension, compared to having valid examples only.

The last null and alternative hypotheses correspond to the second research question. We perform quantitative analysis to accept or 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.

3.4 Problem Domain

We selected rewards loyalty programs as the problem domain of our experiment. Loyalty programs represent organized sales and marketing activities, especially in the retail domain, that reward and hence 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.

3.5 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 model creator were the thesis author and a member of the Generative Software Development (GSD) lab, at the University of Waterloo. They were both familiar with entire experimental setting, and they had advanced knowledge of UML class and object diagrams.

3.6 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 assigned randomly to treatments, but we ensured that for each session we had participants for both treatment

groups. For example, if we had four participants in a session, we printed 2 study materials for the control group, and 2 for the EDM group. Participants would then choose randomly one of the printed materials.

The experiment involved two tasks. **Task 1** asked the participants to create an object diagram for a leading grocery retailer in Canada: Sobeys. The rewards loyalty program for Sobeys is called Club Sobeys. We provided a brief overview about what Club Sobeys is. Creating the object diagram requires understanding of the model abstractions. 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 be included 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.

In order to ensure that any improvement in the EDM group answers over the control group is due to the presence of examples, participants needed to perform more than one comprehension task. **Task 2** of the experiment asked the model receptors to answer a set of 15 study 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. The experimental procedures are displayed in Fig 3.1. From the beginning and throughout the experiment, we encouraged the model receptors to ask any questions they had throughout the whole session. We recorded all questions but we only answered questions related to the tooling or understanding of the requirements.

We used MagicDraw [1] for UML modeling. We introduced the tool during the first 15 minutes of the session and provided training materials on using the tool to open the provided diagrams and create object diagrams.

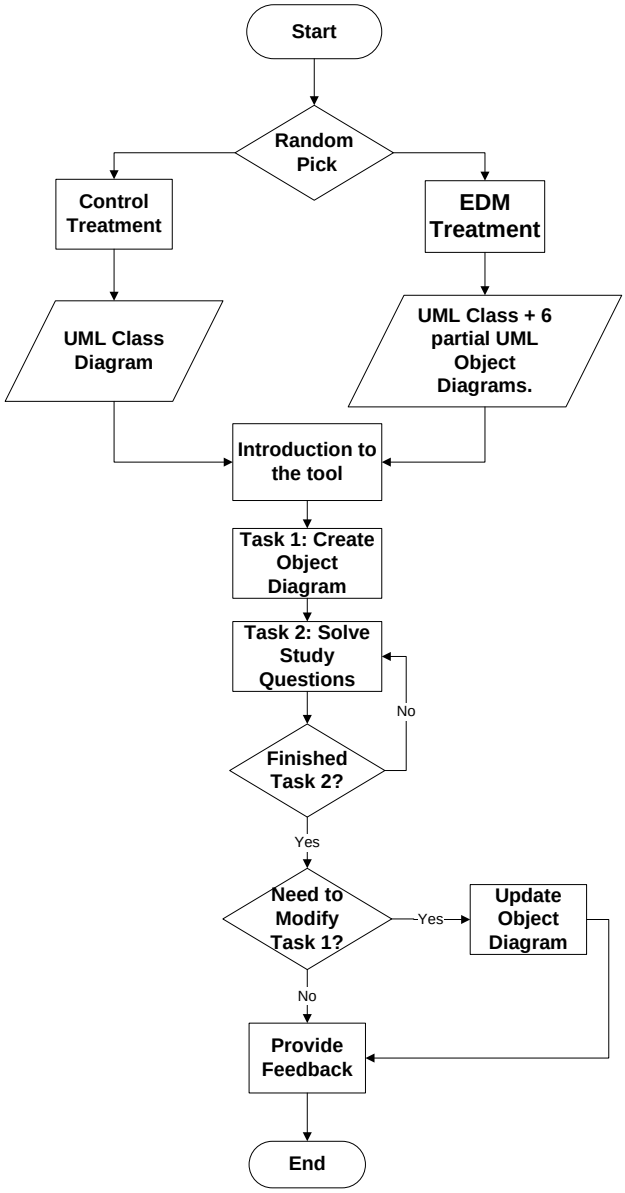


Figure 3.1: Experimental procedures flow chart.

3.7 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 members of the GSD lab at the University of Waterloo, which allowed us to improve the task requirements, enhance the class and object diagrams, and address threats to validity before conducting the experiment on a larger scale. Next and before conducting the first session of the experiment, we conducted a test session to simulate the actual experiment in order to make sure that the requirements were clear and that the tasks were doable in the allocated time. We did not include any of these data points in the analysis. We conducted 10 sessions for the main experiment. Each session had on average two to three model receptors, each assigned to either the control group or the EDM group.

Chapter 4

Data Collection

4.1 Model Abstractions and Examples

We gathered data about loyalty programs in general and two programs in specific: 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 member in the loyalty program. The members earn points for every paid transaction, when they swipe their card upon checkout. The earned points could be base points accumulated through regular transactions and/or bonus points if the store's products are associated with a points offer. A rewards loyalty program generally has partners through which points conversion could occur through a predetermined conversion system. For example, Sobeys might form a partnership with Aeroplan. Aeroplan is a loyalty program aimed at collecting miles to be redeemed for free plane tickets. Club Sobeys members might choose to convert their Sobeys points to their Aeroplan account through a conversion system such as 1 mile for every 2 club sobeys points. All earned points are added to the member's account. Members can then redeem their points via one of the redemption channels provided in the store. An example is the redemption of points for 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) [46]. An extract from the class diagram is shown in Fig 4.1. The full class diagram is available with the rest of experimental materials in Appendix A and Appendix B.

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. A variety of examples were used: four valid and two invalid examples. For instance, example 1, presented in Appendix B, is a valid example that shows how a Shoppers Optimum member earned bonus points for an in-store offer. On the other hand, example 6 shown in Fig 4.2, represents an invalid example that violates a constraint; a member cannot earn and redeem points for the same bill. The full list of examples are attached in Appendix B, and the concept(s) they convey are presented in Table 4.1.

Table 4.1: List of examples for EDM treatment.

Example	**Description**
Example 1	Shows how a member earns bonus points for a *"NiveaBodyCare"* in-store product.
Example 2	Shows how a member earns bonus points through different bonus mechanics for two separate in store products, as well as a regular transaction with no offers for one product.
Example 3	Shows an invalid instantiation of the class diagram since it violates the following constraint: "At most one price or Bonus mechanic per offer".
Example 4	Shows how a member earns bonus points for partner offers.
Example 5	Shows how a member can redeem some of their points.
Example 6	Shows an invalid instantiation of the class diagram since it violates the following constraint: "You can't accumulate and redeem points for the same bill".

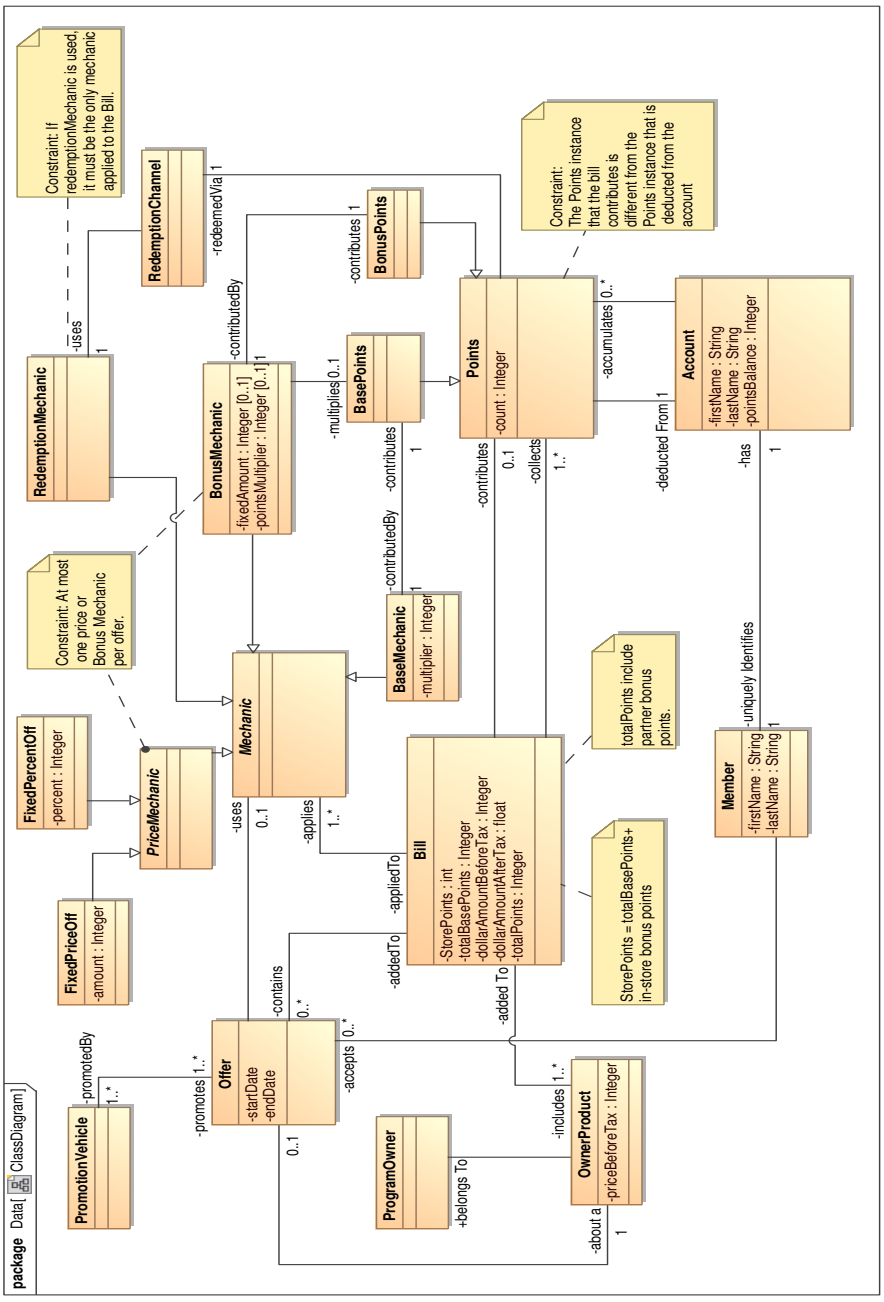


Figure 4.1: UML domain model abstractions.

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 (found in Appendix A). The exercise aimed at making sure that the students rated their experience, 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% or higher in the assessment exercise to be model receptors in our experiment. Among the 26 selected students, only two scored below 80%.

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

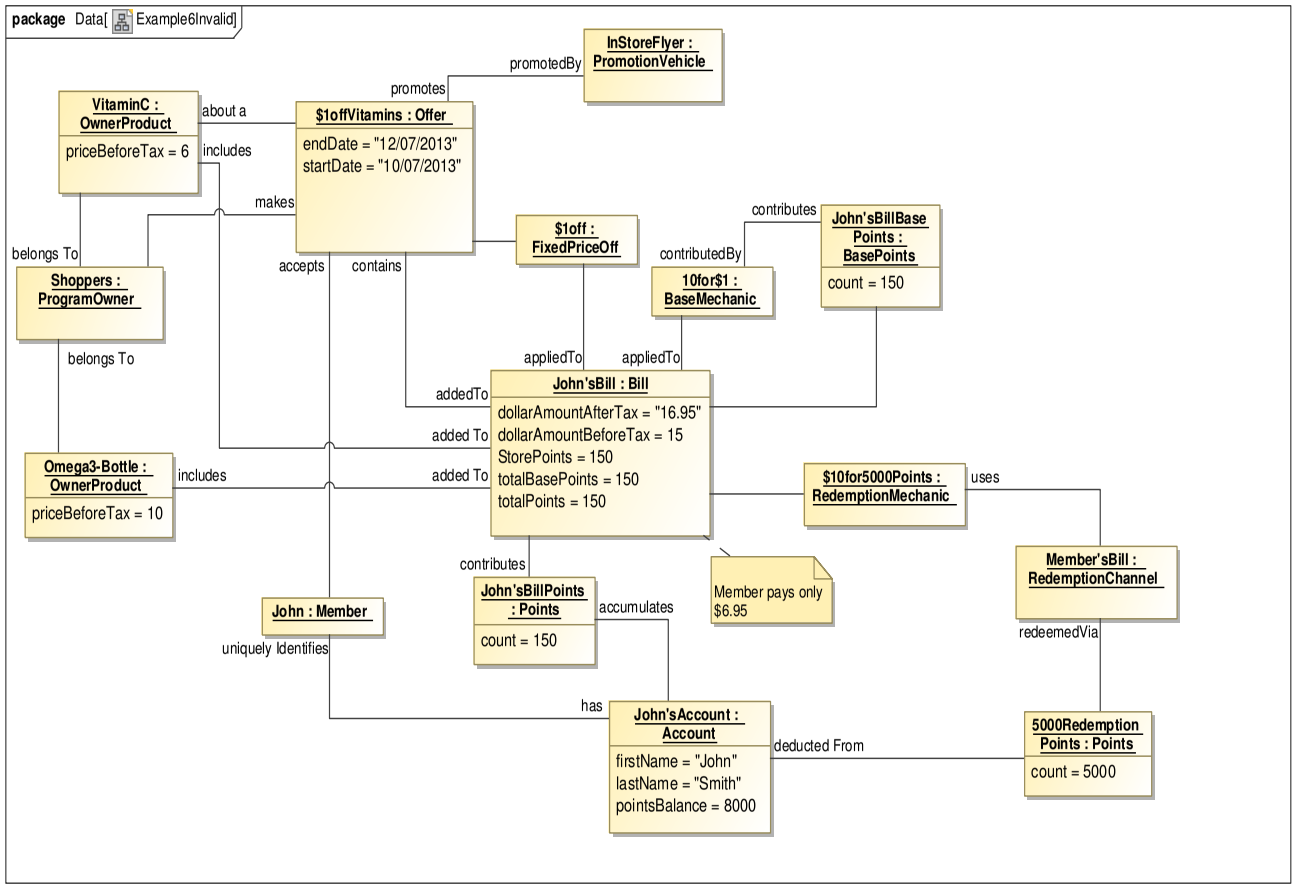


Figure 4.2: Partial object diagram representing an invalid partial example: accumulation and redemption can not be applied for the same bill.

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 once the participants started task 2 until they were done. However, if after the participant finished task 2, they wanted to do some fixes in their object diagram, the experimenter allowed them to do so.

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 another researcher in addition to the employed marking scheme, which allowed him to perform blind verification of the grades. The author and the researcher discussed the differences and agreed on the final grading.

4.5 Participants' Feedback

Each experimental 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.

Chapter 5

Data Analysis

5.1 Preliminary Data Analysis

From the participants' solutions for task 1, we observed that most mistakes and/or missing objects from the control group object diagrams were related to the *RedemptionMechanic*, *RedemptionChannel*, *BonusMechanic*, and *PartnerOffer* classes. All participants from the control group scored zero for the *RedemptionMechanic* class, 11 out of 12 participants scored zero for incorrect or missing instance of the *RedemptionChannel* 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 *PartnerOffer*, the *RedemptionMechanic* classes, and 11 out of 13 participants got the full mark for the *BonusMechanic* and *RedemptionChannel* classes. The inability to correctly instantiate the *RedemptionMechanic* and *RedemptionChannel* classes by the control group indicate 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 a member can redeem their points and yet they were unable to fully comprehend and use that information correctly in their object diagrams: "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 each participant from each treatment answered ***Q9*** correctly. ***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 the EDM chapter 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:*** *"What are the type(s) of points that are collected by the bill?"*, ***Q8:*** *"When would it be the case that a Bill does not contribute Points to the members Account?"*, ***Q10:*** *"Describe in your own words the difference between the "fixedAmount" and the "PointsMultiplier" attribute inside the BonusMechanic class."*.

We observed the highest discrepancy between the scores of the control group was 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 in their object diagrams, 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.

5.2 Outlier Analysis

In order to draw valid conclusions, we followed the suggestion of Wohlin et al. [49] regarding the removal of outliers, caused by exceptional conditions, before applying our statistical tests. During one of the experiment sessions, one participant assigned to the control treatment couldn't finish the experimental tasks within the allocated time. At the end of the three hours, the experimenter marked how many questions the participant solved, but allowed him extra 25 minutes to complete the tasks. The results from this participant were excluded as he solved only 3 questions from task 2 by the end of the three hours.

5.3 Subjects Analysis

We conducted the experiment with 26 participants in several runs. After excluding 1 data point during the outlier analysis, 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.

Chapter 6

Results

6.1 Quantitative Analysis

Since the design of our experiment is 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 the experimental 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 [31]. The assumption was not met for diagram completeness. We used the Welch's t-test for that case, which is an adaptation of the student's t-test intended for use when the two populations have unequal variances [47].

We performed all tests using the R statistical package [2]. 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 6.1.

Table 6.1: 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

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.

On average, an EDM participant scored 88% for diagram completeness compared to an average of 49% for a control participant. This is a 39% improvement. Given that a participant requires model and domain understanding to create the object diagram, these results indicate a 39% comprehension improvement when the model abstractions are presented together with examples. Figure 6.1 shows an example of a miss in an object diagram. The object diagram is missing the *BasePoints* instance; when *BaseMechanic* is applied, it contributes *BasePoints*. These points are then collected by the *Bill*. The diagram is also missing the attribute values from the *Offer* instance.

Applying our statistical tests, we saw a significant effect of examples on diagram completeness, with a *p-value* less than 0.0002, indicating that the mean completeness score for the EDM group was significantly higher than the mean completeness score for the control group. Moreover, one interesting observation, as shown in Figure 6.2, is that the minimum diagram completeness percentage (81.2%) scored by an EDM participant was higher than the maximum completeness percentage scored by a control participant (69.6%).

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, associ-

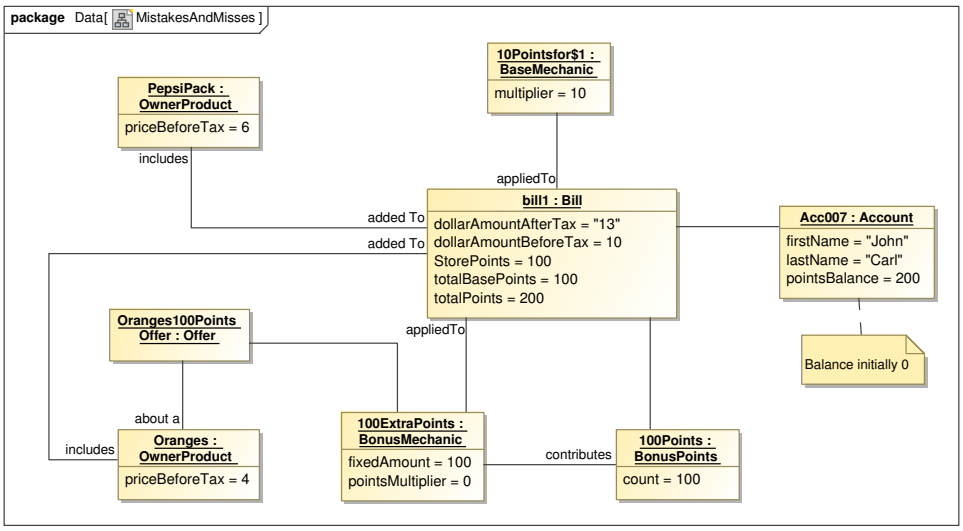


Figure 6.1: Sample solution showing mistakes and misses similar to those done by the actual participants.

ations, 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 *Bill* instance, 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. Detailed analysis of the missing objects from the participants' solutions as well as the participants' mistakes are provided in Appendix C.

We created an example of two mistakes that we found in the control participants' diagrams, shown in Figure 6.1. The first mistake is in the *Bill* instance. According to the points calculation rule presented in the class diagram, the *StorePoints* attribute should have the value 200 since it is the sum of base and bonus points accumulated from in-store products. Additionally, since there are no points contributed by a partner, the *totalPoints* attribute would be equal to 200 as well. The second mistake is the direct association between the *Bill* and *Account* instances. The *Bill* should first contribute a *Points* instance, which is then accumulated by the *Account*. There shouldn't be a direct link between the *Bill* and *Account* instances in the object diagram.

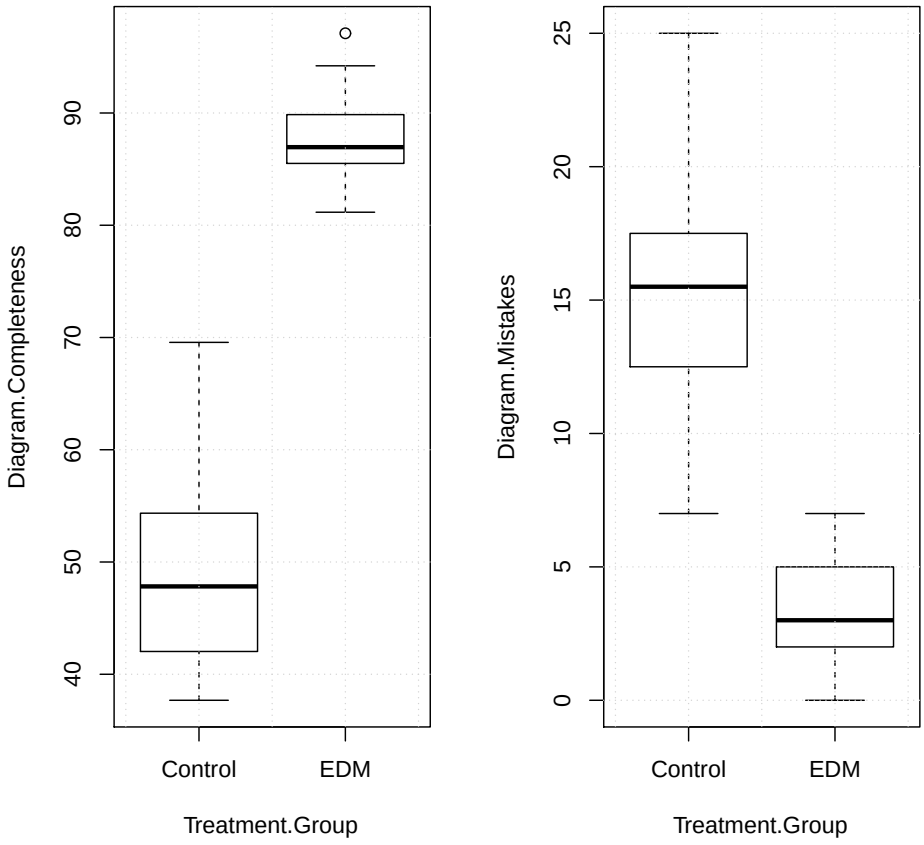


Figure 6.2: Diagram Completeness and mistakes.

On average, an EDM participant had 3 mistakes in his or her object diagram compared to an average of 15 mistakes for a given control participant's diagram. This is almost an 80% reduction in the number of mistakes. The data illustrated in Figure 6.2, with a *p-value* less than 0.0053, 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. Additionally, our results show that the maximum number mistakes (7 mistakes) found in an EDM participant's diagram is equal to the minimum number of mistakes found in a control participant's diagram.

Task 2 Completeness. This variable measures each *participant's score for answering the study 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 Appendix A. 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.

We observed an expected significant effect of examples on the participants' solutions for task 2, with a *p-value* less than 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 6.3. If on average an EDM participant scored 21 marks for task 2 completeness compared to only 13 marks for a control participant, this would be an average improvement of almost 30% for task 2 completeness due to the presence of examples.

Efficiency. In our analysis, we use the 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. 6.3 shows the efficiency scores for the EDM group compared to the control group. Comparing the average efficiency score for an EDM participant of (1.2) to that of a control participant (0.7), we can see almost 70% efficiency improvement. With a *p-value*=0.00082, the results show that there was a significant effect on improving the efficiency scores for the EDM group compared to the control group.

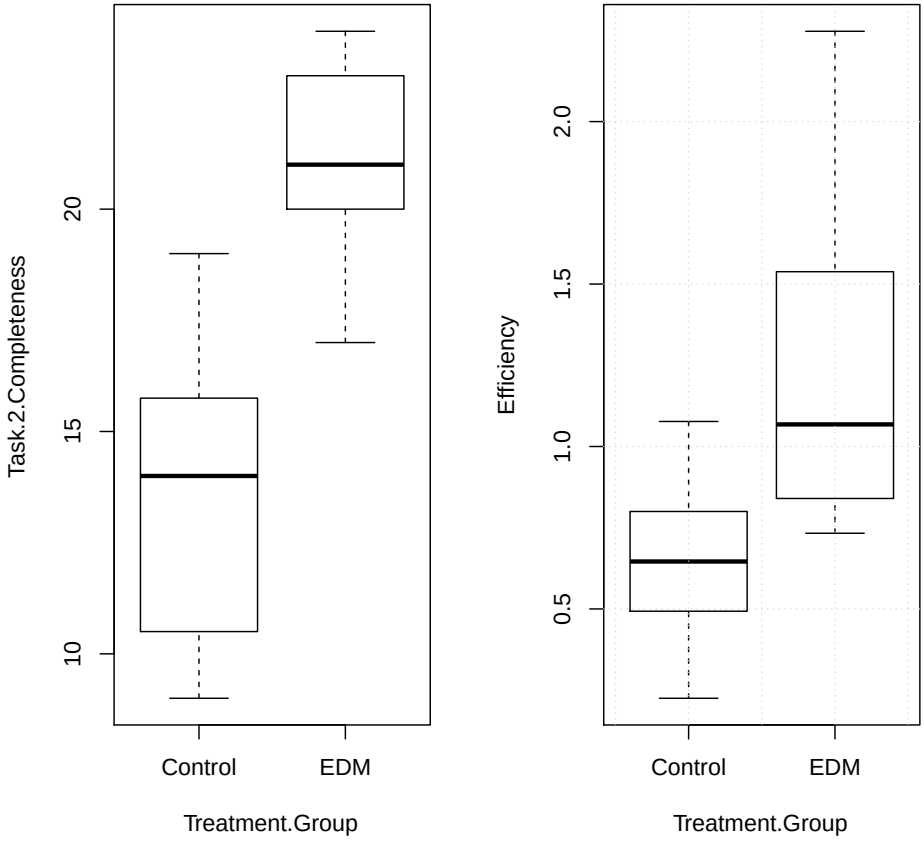


Figure 6.3: Task 2 completeness and efficiency.

6.2 Qualitative Analysis

We present our qualitative analysis in terms of the participant's feedback provided at the end of each experimental session. First, we asked participants from both groups to rate the difficulty level of their experience in performing the comprehension tasks: (1) in creating the object diagram and (2) 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 compared to an average of 3.83 by the control group. This represents an average decrease of 27% in the participants' perceived difficulty for creating the object diagram when they have the examples together with the model abstractions. EDM participants rated themselves on average 2.08 for difficulty in answering task 2 questions, while participants from the control group rated their experienced difficulty an average of 3.83. This difference is almost 24% decrease in the participants' perceived difficulty for answering task 2 questions.

These results are in alignment with the feedback we got from the EDM participants. For example, they all agreed that it would have been hard to understand the class diagram without examples. One participant's response was *"The class diagram was huge, I didn't know where to start reading to understand, so I looked at the examples one by one while looking at the model abstractions simultaneously"*. The difference in the participants' experienced difficulty was also apparent in the number of questions about the domain (i.e. the UML class diagram) raised by the EDM versus the control groups. On average, a control group raised 10 domain-related questions per experimental session, compared to only one domain-related question by the EDM group. This is an average decrease of 90%. Examples of domain-related questions and their relative frequency are shown in Table 6.2.

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*, "Partner" *Offer*, and *Bill*. The EDM group also mentioned that the *BonusMechanic*, *RedemptionMechanic*, and *Bill* classes were hard to comprehend from the abstractions only, but the examples helped a lot. As for the most difficult associations, these were the associations related to these classes. Table 6.3 shows the number of mistakes and misses for the above mentioned classes in the control participants' diagrams versus the EDM participants' diagrams. Note that the class *Offer* is used once for an *OwnerProduct* and another for a *PartnerProduct* and hence the name "Partner" *Offer*.

Since both control and EDM groups agreed that these classes were hard to understand, we can attribute the decrease in the number of mistakes and misses for these classes in

Table 6.2: Domain-related questions raised by participants.

Question	**EDM Frequency**	**Control Frequency**
What do you mean by PromotionVehicle?	0	8
Why do we need OwnerProduct and PartnerProduct?	0	6
What is the difference between RedemptionMechanic and RedemptionChannel?	1	11
What is the difference between storePoints and totalPoints attributes in the Bill class?	2	9

Table 6.3: Classes that were most difficult to comprehend.

Class Name	**Number of Mistakes**		**Number of Misses**	
	EDM	Control	EDM	Control
BonusMechanic	2	5	0	4
RedemptionMechanic	0	9	0	3
"Partner" Offer	0	6	0	3
Bill	3	9	0	0

the EDM answers to the presence of examples. One interesting observation based on the participants' solutions is that they didn't mention the *RedemptionChannel*, *OwnerProduct*, and *Points* classes as hard to understand although they had a similar number of mistakes and/or misses. For example, all control participants got zero for both *RedemptionMechanic* and *RedemptionChannel* classes, whether due to mistakes or missing instances in their object diagrams.

Some of the reasons the control group found these classes and associations difficult to understand 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."*, *"the concepts for these classes unlike Member or Account classes don't hold any previous*

meaning and hence were difficult to get their idea", and finally *"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."*

We asked the EDM participants to list the examples they found most useful and helpful. The majority mentioned examples 2, 4, 5, and 6. 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 points attributes from different classes relate and the sequence for accumulating points. For example, you have to calculate in-store points first before you apply partner points"*, *"example 4 shows that the account points balance gets updated 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 system behavior presented in example 6 was 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, incorrectly answered the question in task 2 related to the constraint violation illustrated by example 6. The question was: *"Can a customer earn and redeem points for the same bill?"*. He replied with "yes" although the correct answer is "No". 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 as well as build a better mental model.

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 participants rated their confidence level on average 4.08 for their object diagram answers compared to 2.33 by the control group. This represents an average improvement of 35%. EDM participants rated their confidence level on average 4.15 for their task 2 answers, compared to an average confidence level of 3.00 for the control group. This represents a 23% average increase in confidence for 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.

Chapter 7

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 best supports. 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. This feedback suggests some recommendations for using examples such as: (1) a variety of examples is needed when there is more than one association linked to a single class. In other words, when you have a domain concept with several possible usage alternatives. This could be more than one valid examples, each representing an alternative or one valid and one invalid examples demonstrating the correct system behavior through contrasting representations. (2) valid examples are needed to show the propagation of attributes through classes. This was of high importance for the process of points accumulation where you calculate the base and bonus points for each product, add them to the bill, then calculate partner points (if any). The total number of in-store and partner points account for the total accumulated points per transaction to be added to the account. (3) valid examples are needed with unfamiliar domain concepts such as with the classes representing the process for points redemption. (4) and finally, invalid examples are needed to ease the comprehension of complex constraints.

The inability of the control group to correctly instantiate and use the *RedemptionMechanic* and *RedemptionChannel* classes shows the usefulness of explicit examples over any

Table 7.1: One instance of the redemption process description as a check list item.

Possible ways by which a member can redeem points previously added to their account such as: (1) Inside the store to reduce overall grocery costs. (2) Through a Club Sobeys partner (For example; Aeroplan. Aeroplan loyalty program was created in July 1984 by Air Canada as an incentive program for its frequent flyer customers. Aeroplan members collect miles via credit cards or through a conversion system between their partners such as Sobeys.)

other form or representation of extra information. We included a textual description of the redemption process and possible ways through which members can redeem their points. One example of this description was presented as a check list item that participants needed to include in their object diagram. This item is given in Table 7.1.

The participants' solutions to the comprehension tasks suggested a preliminary indication about the type of stakeholders that would benefit from our approach: *Examples are mostly needed by stakeholders who are novices to the application domain.* Question 9 in task 2, as previously mentioned, asked the participants to describe the difference between *FixedPriceOff* and *FixedPercentOff* mechanics. All control participants answered this question correctly, probably due to previous shopping experience. This suggests that when model receptors are familiar with or experts in the domain, they wouldn't need examples to ease the comprehension process.

Through our detailed task analysis, we wanted to see whether the effect of examples on task 2 completeness was independent of 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 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.

Chapter 8

Threats to Validity

This section discusses the threats to internal, external, construct, and conclusion validity according to [49, 20].

8.1 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 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 concerned 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 in Task 2 questions targeted this activity. Moreover, 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.

8.2 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 industry practitioners.

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.

8.3 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' rating of their expertise in UML class and object modeling. However, in order to make sure we accurately measured the students' background, we sent them an additional 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.

8.4 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.

Chapter 9

Related Work

9.1 The Use of Examples in Creating Domain Model Abstractions

The idea of abstracting a set of domain examples into a more general model is not new. In [10], the authors propose a framework for domain specific modeling language (DSML) creation. The idea is that the SMEs provide a set of domain examples that are later transformed into graph representations. The framework uses a graph builder to transform the domain examples, provided by the end-users, into a set of graphs. Then, the concrete syntax identifier identifies visual modeling elements from the graph representations as the candidate concrete syntax. This candidate concrete syntax is provided to the end-users, who are domain experts, to be reviewed and annotated. During review and annotation, the end-users may be asked to provide extra information related to the association links between the model elements if needed. "After the concrete syntax is specified, the Graph Annotator rewrites the graph representations generated by the Graph Builder with the names of concrete syntax" [10]. The Graph Annotator completes the graphical model generation through graph structure optimization to remove any unneeded complexity. The framework then uses its Metamodel Inference Engine that infers abstract syntax based on the graph representation with the concrete syntax. Metamodel inference in this framework represents a form of inductive learning since the output Metalmodel is induced by learning from examples [26]. The final step is inferring the semantics from the domain examples.

The major challenge of this approach is that it requires the end-users to represent the domain examples in the same modeling language as the one intended for representing the

general model. For example, the SMEs would provide example finite state machines if they require a general finite state machine. The approach ignores the fact the most of the domain experts are only familiar with their domain and are not experts with specific modeling languages. The framework also supports positive examples only which makes the process of inferring the Metamodel longer and more error-prone since one might miss important constraints that need to be included in the model.

9.2 UML Model Comprehension

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 [50] tests the effect of different layouts strategies on UML model comprehension. The results indicate improved accuracy in solving comprehension tasks when having clustered layouts. The experiment replication presented in [41], uses more number of participants and compares different layout strategies that are based on class stereotypes: entity, control, and boundary. Results show improved participants' performance in answering UML syntax and software design questions when having clustered layouts. Similarly, the experiment presented in [29] 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. In [36], the paper investigates whether different notational variations for UML class diagrams have different effects on model comprehension. The authors used two notations that were semantically equivalent, but syntactically different. The results show that the nature of the task determines the best performing notation with regards to comprehension.

9.3 Effects of Using Object Diagrams in UML Class Diagram Comprehension

Although there exist a lot of 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 improving model comprehension. The first study in [42], 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 [39] 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, but achieves 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.

Chapter 10

Conclusions

10.1 Summary of Findings

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 (+31%) 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 (24 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 confidence increase in their created diagrams (+35%),

and 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. As for having a variety of valid and invalid examples compared to having valid examples only, 12 out of 13 EDM participants preferred having them, while the only participant who said that valid examples only are enough answered the study question related to one of the constraints incorrectly. We cannot depend on this data to reject the last null hypothesis. However, it provides insight on our future directions.

Our analysis suggests that examples are needed the most when the concepts of the model abstractions are not familiar to the model receptor (i.e. when the model receptors are novices to the application domain). 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. 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 [5, 6, 4] and showed how it could be used for EDM [3].

Finally, we designed our experiment with replication in mind. We included the supplementary materials, 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, in Appendix A and B. 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.

10.2 Limitations and Future Work

During our feedback sessions, all EDM participants preferred having partial examples to having one complete object diagram. However, one participant mentioned that having one complete object diagram in addition to the partial examples would have helped him more in

forming a complete picture and improving his understanding of the domain. This suggests a direction for future work, that 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 EDM participants also commented on the usefulness of specific examples. This opens some interesting opportunities in identifying not just guidelines for using EDM, but some actionable knowledge as well. We can have EDM patterns whose usage is determined based on the aspects of the domain being modeled. For example, if you have more than one association for a single class (concept), cycles or deep hierarchies in your model abstractions, then you would probably need specific types of examples. Also, although we only used UML class and object diagrams to represent structural model abstractions and their examples, we argue that our technique is independent of the notation used to represent the structural model as long as it has the capability of representing both the model abstractions and examples.

Finally, in this experiment we had one independent variable (modeling method) with only two treatments (model abstractions only, and model abstractions+a variety of examples). For future work, we need to also evaluate the effects of using a variety of valid and invalid examples as compared to having valid examples only. The value added to model and domain comprehension from having a variety of examples compared to having valid examples only might not be significant when taking into account the cost of creating and maintaining these invalid examples. This also applies to the cost of having examples in general in addition to the model abstractions. Therefore, for future work we need to also verify the usefulness and the cost-benefit ratio of applying EDM, but in an industrial context.

APPENDICES

Appendix A

Study Materials

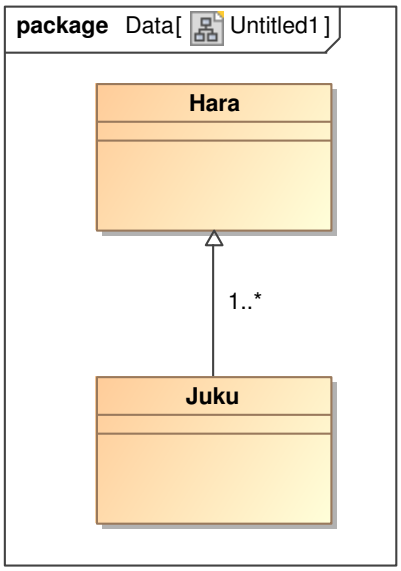
We took some questions presented in the UML knowledge assessment exercise from the online sample practice tests for the Sun Certified Java Associate exams.

A.1 UML Knowledge Assessment Exercise

Question 1
What type of relationship is needed to represent the relationship between students and the courses they are enrolled in at a university?

1. A one-to-one association.
2. A one-to-one composition.
3. A one-to-many association.
4. A one-to-many composition.
5. A many-to-many association.
6. A many-to-many composition.

Question 2
Exhibit:



Which of the following is true?

1. Juku is a subclass of Hara.
2. This is NOT a valid UML class diagram.
3. Every Juku has a reference to at least one Hara.
4. Juku is a subclass of Hara and at least one other class.

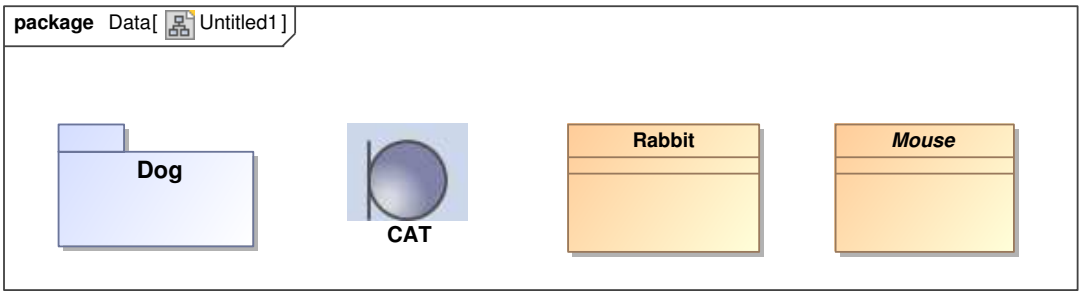
Question 3
Which two are true about composition relationships? (choose two)

1. Composition relationships can be one-to-many.
2. Composition relationships are never one-to-many.
3. Composition relationships are always many-to-many.
4. Composition relationships are used to show exclusive ownership.

Question 4
Which two are true about the relationship "A keyboard has 101 keys"? (choose two)

1. This is a one-to-one relationship.
2. This is a composition relationship.
3. This is a one-to-many relationship.
4. This is a many-to-many relationship.
5. This is not a composition relationship.

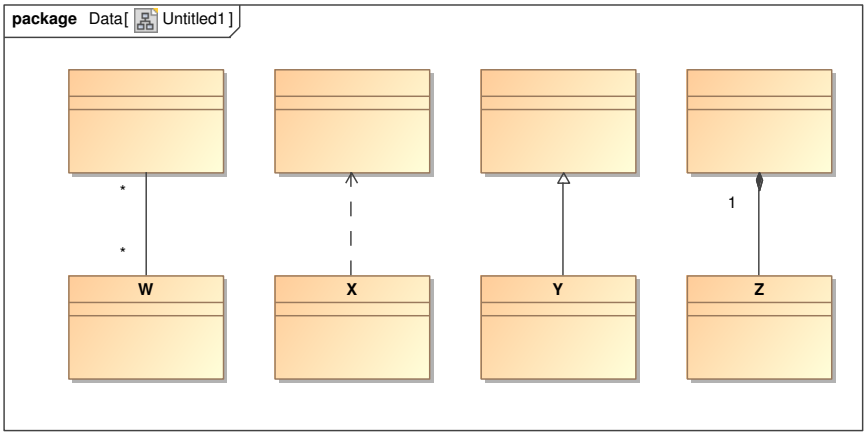
Question 5
Exhibit:



Which is an abstract class?

1. Cat.
2. Dog.
3. Rabbit.
4. Mouse.

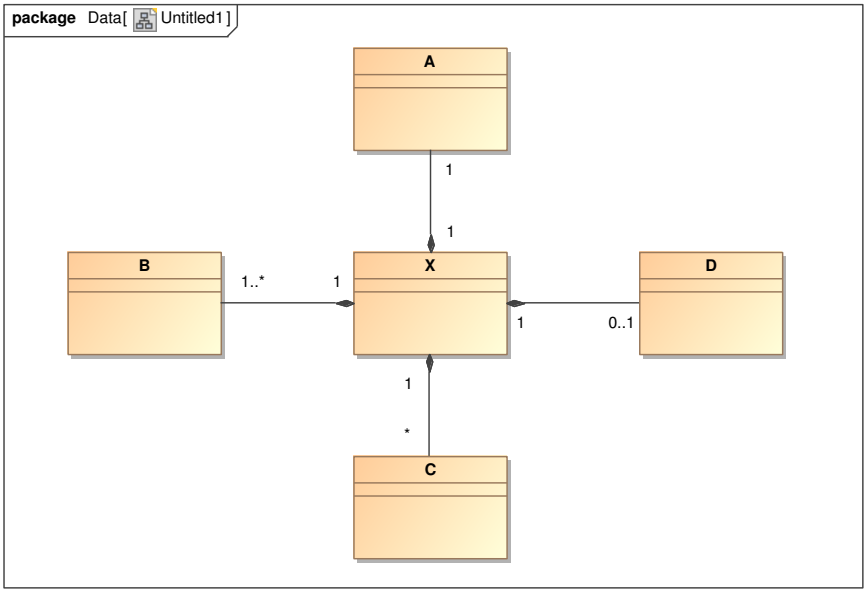
Question 6
Exhibit:



Which class has a superclass relationship?

1. W.
2. X.
3. Y.
4. Z.

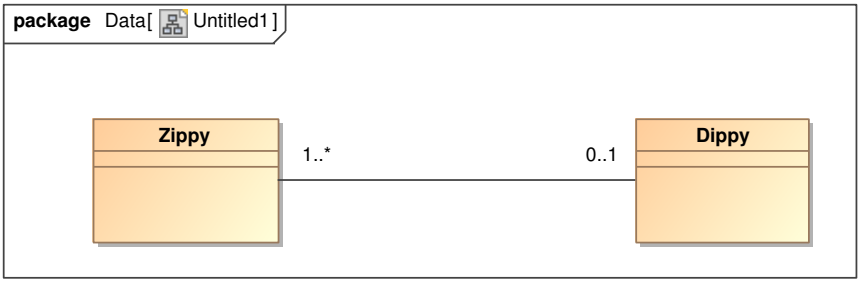
Question 7
Exhibit:



Which two classes can have two or more instances associated with a single instance of X ? (choose two)

1. A.
2. B.
3. C.
4. D.

Question 8
Exhibit:



Which two are true? (choose two)

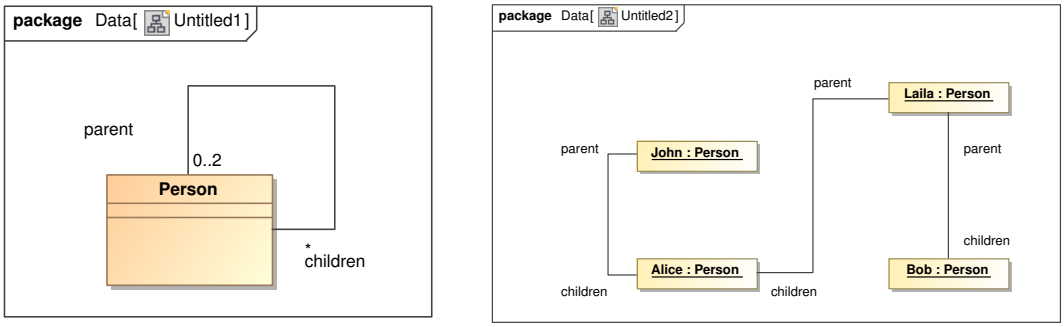
1. It is valid for a Zippy to have no associated Dippy.
2. It is valid for a Dippy to have no associated Zippy.
3. Every Zippy must be associated with exactly one Dippy.
4. Every Dippy must be associated with exactly one Zippy.
5. Every Dippy must be associated with at least one Zippy.
6. It is valid for a Zippy to be associated with more than one Dippy.

Question 9
Which two are true? (choose two)

1. 2..4 is a valid multiplicity indicator..
2. The multiplicity indicators * and 1..* are equivalent.
3. The multiplicity indicators + and 1..* are equivalent.
4. An optional association is shown using the multiplicity indicator 0..1.
5. Multiplicity indicators must always be shown on both ends of an association.
6. Multiplicity indicators are optional, but if they are included they must be shown on both ends of an association.

Question 10
Model the relationship between a car (that has an engine and color) and its owners (having a name) in a UML class diagram. A car can have several owners over time, but only one owner at a time. Do not forget cardinalities, role names, attributes, and their types. Note: An engine cannot exist without a car.

Question 11
Exhibit:



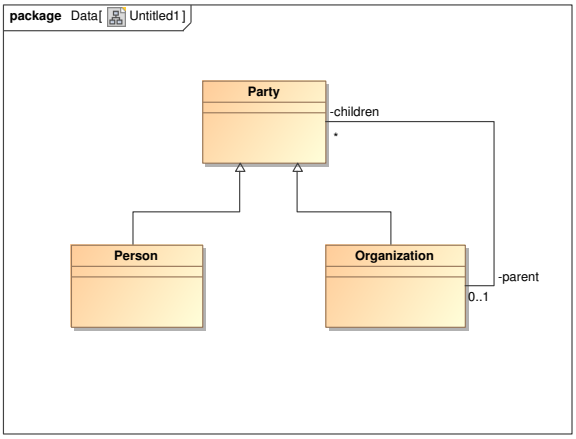
(a) UML class diagram

(b) UML object diagram

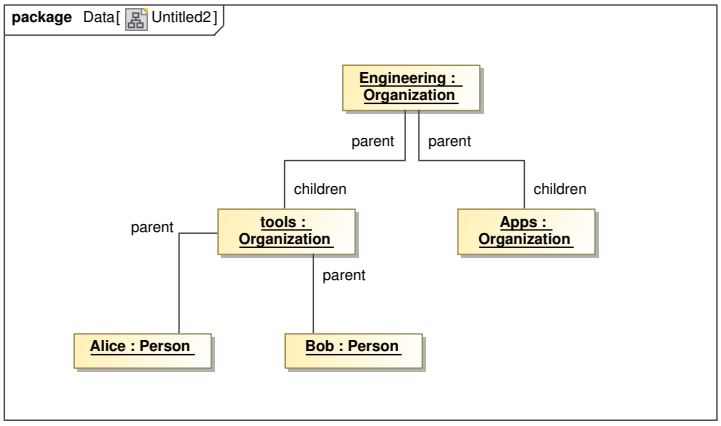
Does the object diagram (Fig. A.1(a)) represent a valid instantiation of the class diagram (Fig. A.1(b))?

1. Yes.
2. No.

Question 12
Exhibit:



(c) UML class diagram



(d) UML object diagram

Is Figure A.1(d) a valid instantiation of Figure A.1(c)?

1. Yes.
2. No.

If your answer to the previous question is No, please provide a correct instantiation from your point of view.

A.2 Study Materials given to the Control Group

A.2.1 Info Page

Hello, My name is Dina Zayan. I am the student investigator for the Structural Modeling study you agreed to participate in. The study will take place over a single 3-hour session. During the 3-hour period, you will be asked to sign a consent form, read a UML class diagram, create a UML object diagram as well as answer a set of questions. Feel free to ask any questions if you need during the study. You are free to withdraw from the study at any time. I would like to thank you in advance for your time and effort.

Regards,
Dina Zayan

A.2.2 Consent

I have read the information presented in the recruitment letter about the study being conducted by Dina Zayan and Associate Professor Krzysztof Czarnecki at the University of Waterloo.

I have the opportunity to ask any questions related to this study, to receive satisfactory answers to my questions, and any additional details I want. I am aware that I may withdraw from the study without penalty at any time by advising the researchers of this decision.

This project has been reviewed by, and received ethics clearance through, the Office of Research Ethics at the University of Waterloo.

With full knowledge of all foregoing, I agree, of my own free will, to participate in this study.

Participant's Name:

Participant's Signature:

Date:

Witness Name:

Witness Signature:

A.2.3 Experimental Procedures

Club Sobeys is a program designed to reward members who shop at Sobeys, a leading national grocery retailer and food distributor in Canada. Every time a member swipes their card during checkout, they get rewarded using the programs points system. A member can redeem their points for instant savings or have them converted automatically to one of Sobeys partners. Please exhibit the systems attached UML class diagram.

Experimental Task 1
Please create a UML Object Diagram which represents a valid concrete instantiation of the attached UML Class Diagram for a rewards program for Club Sobeys members.

When creating your object diagram, make sure the following list is present in your instantiation:

Item	Check Box
An account that uniquely identifies each member.	
A member earning points for regular transactions (i.e. no offers on the products).	
A member earning points through in store offers (i.e. Offers on products by the store).	
A member earning points through partner offers. For example, *BMO/Club Sobeys MasterCard* which is a product offered by one of Club Sobeys partners: BMO Bank.	
Sales channels through which the customer/member will know about the offers taking into account different ways for promoting in store and partner offers.	
The object diagram must include at least one instance of each class in the class diagram.	
All attribute values of the instances in the object diagram must be included.	
Possible ways by which a member can redeem points previously added to their account such as: • Inside the store to reduce overall grocery costs. • Through a Club Sobeys partner (For example; Aeroplan. Aeroplan Loyalty program was created in July 1984 by Air Canada as an incentive program for its frequent flyer customers. Aeroplan members collect miles via credit cards or through a conversion system between their partners.)	

Experimental Task 2

Please provide answers for the following questions based on the first task of the study. Try to answer each question; however, if you are not sure and feel you are guessing, do not provide an answer.

Please record the time when you start and finish answering the questions as indicated.

Start Time: .....

1. Which of the following answers is true?

   (a) Partner offers are applied to the entire members bill.

(b) Partner offers can be applied to individual products inside the store.

(c) Both answers are correct.

2. Can a customer earn and redeem points for the same bill?

   (a) Yes.

   (b) No.

3. Is *Base Mechanic* mandatory in the case of earning points?

   (a) Yes.

   (b) No.

4. Are the *Points* collected by the *Bill* those that are deducted from the *Account*?

   (a) Yes.

   (b) No.

   (c) Can't be determined.

5. What are the type(s) of points that are collected by the bill?

6. Are Bonus and/or Price mechanics mandatory?

   (a) Yes.

   (b) No.

7. When can a **Bill** have "BonusMechanic" without "BaseMechanic"?

8. When would it be the case that a *Bill* **does not** contribute *Points* to the members *Account*?

9. Please provide an example to demonstrate the difference between a *"FixedPriceOff"* mechanic and a *"FixedPercentOff"* mechanic.

10. Describe in your own words the difference between *"fixedAmount"* and the *"Points-Multiplier"* attribute inside the *"BonusMechanic"* class.

11. Which class(es) represent different ways of collecting points?

12. Why do you think the *"BasePoints"* class is associated with the *"BonusMechanic"* class?

13. What is the conceptual difference between the *"StorePoints"* and the *"TotalPoints"* attributes inside the *"Bill"* class?

14. Is it possible to have both *"Price Mechanic"* and *"Bonus Mechanic"* in the same object diagram? If yes, briefly describe the case.

15. Given your created object diagram, assume Sobeys suddenly decides to change its policy to 500 Club Sobeys points for every $1 spent in the store. Please mention **all** instances in your object diagram which would have their values changed accordingly to accommodate the change in Sobeys policy?

End Time: .....

A.3 Study Materials given to the EDM Group

The EDM participants received the same study materials as the control group except for the six partial examples shown in Appendix B, and their description presented in Table 4.1.

A.4 The De-briefing Questionnaire

- How would you rate your experience with creating the object diagram?(1 being the lowest and 5 being the highest)
    1. Very Easy.
    2. Easy.
    3. Moderate.
    4. Difficult.
    5. Very Difficult.
- How would you rate your experience with answering the domain questions in Task 2?
    1. Very Easy.
    2. Easy.
    3. Moderate.
    4. Difficult.
    5. Very Difficult.
- Do you think it would have been hard to understand/comprehend the class diagram without examples?
    1. Yes.
    2. No.
- How would you rate your confidence level in the answers you provided for the object diagram?
    1. Unsure.

2. Not sure enough.
3. Neutral about my answers.
4. Sure.
5. Very Sure.

- How would you rate your confidence level in the answers you provided for the questions in Task 2?

1. Unsure.
2. Not sure enough.
3. Neutral about my answers.
4. Sure.
5. Very Sure.

- Which parts (classes and/or associations) of the class diagram were hard to understand/comprehend without examples?
- Which examples were of particular help to you (if any)?
- Did help seeing a variety of examples (valid and invalid) instead of valid examples only? Why?
- Do you prefer partial examples to convey a concept over one big but complete object diagram? Why?
- Do you have any comments on the experiment?

A.5 The UML Knowledge Assessment Exercise Marking Scheme

Question Number	**Answer**
Question 1	(5) A many-to-many association
Question 2	(2) This is NOT a valid UML class diagram.
Question 3	(1) Composition relationships can be one-to-many, and (4) Composition relationships are used to show exclusive relationships.
Question 4	(2) This is a composition relationship, and (3) This is a one-to-many relationship.
Question 5	(4) Mouse.
Question 6	(3) Y.
Question 7	(2) B, and (3) C.
Question 8	(1) It is valid for a Zippy to have no associated Dippy, and (5) Every Dippy must be associated with at least one Zippy.
Question 9	(1) 2..4 is a valid multiplicity indicator, and (4) An optional association is shown using the multiplicity indicator 0..1
Question 10	More than one answer were accepted.
Question 11	(1) Yes.
Question 12	(1) Yes.

A.6 The Experimental Tasks Marking Scheme

Task 1

Class Name	**Weight(marks)**
Program Owner	1
Owner Product	2
Partner	1
Partner Product	1
Owner Offer	2
Partner Offer	1
Promotion Vehicle	2
Fixed Price Off	2
Fixed Percent Off	2
Base Mechanic	2
Base Points	2
Bonus Mechanic	2
Bonus Points	2
Bill	5
Member	2
Registration	1
Account	2
Card	1
Points (Accumulated)	2
Points (Redeemed)	2
Redemption Channel	1
Redemption Mechanic	1
Regular Product	1

Association Name	**Weight(marks)**
PartnerProduct belongs to Partner	1
OwnerProduct belongs to ProgramOwner	1
Offer about a PartnerProduct	1
Offer about an OwnerProduct	1
Offer promotes/promoted By PromotionVehicle	1
Member accepts an Offer	1
Member performs Registration	1
Registration creates an Account	1
Member has/uniquely identifies Account	1
Account is associated with Card	1
Bill includes OwnerProduct	1
Offer uses BonusMechanic	1
Offer uses FixedPriceOff	1
Offer uses FixedPercentOff	1
Offer addedTo Bill	1
Bill applies BaseMechanic	1
Bill applies BonusMechanic	1
Bill applies FixedPriceOff	1
Bill applies FixedPercentOff	1
Bill applies RedemptionMechanic	1
BaseMechanic contributes BasePoints	1
BonusMechanic contributes BonusPoints	1
Bill contributes Points	1
Bill collects Base Points	1
Bill collects Bonus Points	1
Account accumulates Points	1
Points deductedFrom Account	1
Points redeemedvia RedemptionChannel	1
RedemptionChannel uses a RedemptionMechanic	1

Task 2

1. **Which of the following answers is true? (1 mark)**
   Ans: a) Partner offers are applied to the entire member's bill.

2. **Can a customer earn and redeem points for the same bill? (1 mark)**
   Ans: b) No.

3. **Is BaseMechanic mandatory in the case of earning points? (1 mark)**
   Ans: a) Yes.

4. **Are the Points collected by the Bill those that are deducted from the Account? (1 mark)**
   Ans: b) No.

5. **What are they type(s) of Points that are collected by the Bill? (2 marks)**
   Ans: Base Points and Bonus Points.

6. **Are Bonus and/or Price mechanics mandatory? (1 mark)**
   Ans: C) No.

7. **When can a Bill have "BonusMechanic" without "BaseMechanic"? (1 mark)**
   Ans: a) Not applicable , or b) FixedAmount in Bonus Mechanic is used instead of PointsMultiplier.

8. **When would it be the case that a Bill does not contribute Points to the members Account? (1 mark)**
   Ans: In case of redemption.

9. **Please provide an example to demonstrate the difference between a "FixedPriceOff" mechanic and a "FixedPercentOff" mechanic. (2marks)**
   Ans: Fixed Price off: $2 off a product, while Fixed Percent off: 2% off a products price.

10. **Describe in your own words the difference between "fixedAmount" and the "PointsMultiplier" attributes inside the "BonusMechanic" class? (2 marks)** Ans: Fixed Amount: fixed amount of bonus points (500 points), Points Multiplier: 5 times the base points.

11. **Which class(es) represent different ways of collecting points? (2 marks)**
    Ans: Base Mechanic and Bonus Mechanic, or Base Points and Bonus Points.

12. **Why do you think the "BasePoints" class is associated with the "Bonus-Mechanic" class? (1 mark)**
    Ans: In case of Points Multiplier where the total of bonus points is x times the base points.

13. **What is the conceptual difference between the "StorePoints" and "Total-Points" attributes inside the Bill class? (2 marks)**
    Ans:
    StorePoints: In-store base and bonus points.
    TotalPoints: Store Points and Partner points.

14. **Is it possible to have both "PriceMechanic" and "BonusMechanic" in the same object diagram? If yes, briefly describe the case. (1 mark)**
    Ans: Yes, if they are for different products.

15. **Given your created object diagram, assume Sobeys suddenly decides to change its policy to 500 Club Sobeys points for every $ 1 spent in the store. Please mention all instances in your object diagram which would have their values changed accordingly to accommodate the change in Sobeys policy? (5 marks)**
    Ans:

    1) Base Mechanic.
    2) Base Points.
    3) Bill.
    4) Points (Accumulated).
    5) Account.

Appendix B

Experiment Diagrams

B.1 Domain Abstractions (UML Class Diagram)

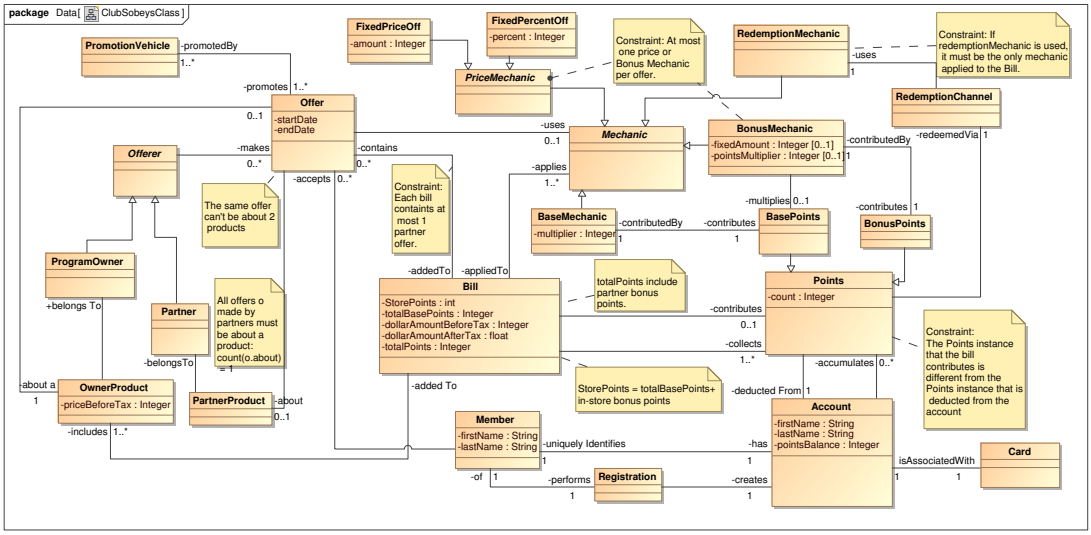


Figure B.1: Rewards loyalty programs' class diagram.

B.2 Domain Examples (UML Object Diagrams)

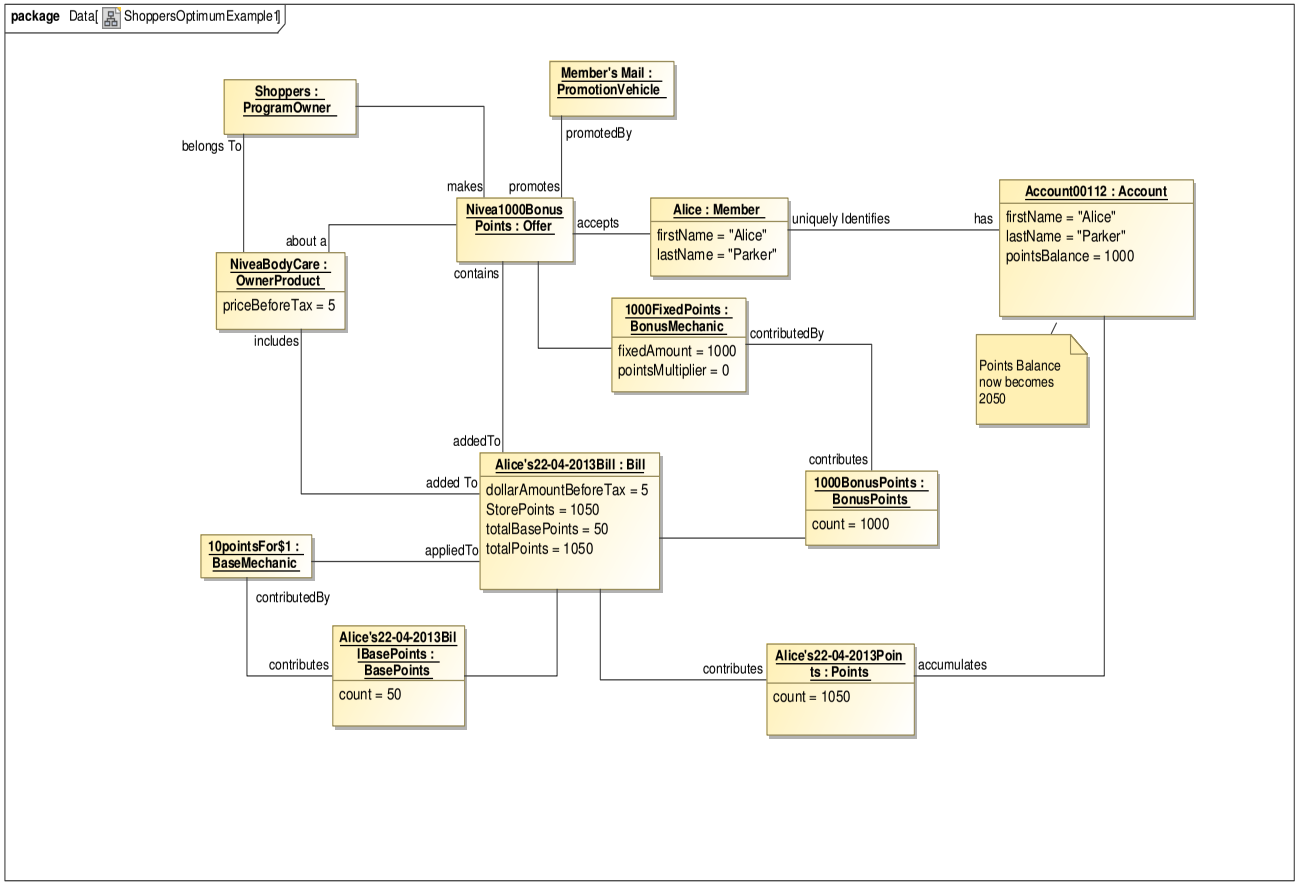


Figure B.2: Example 1 showing how a member earns bonus points for a *NiveaBodyCare* in store product.

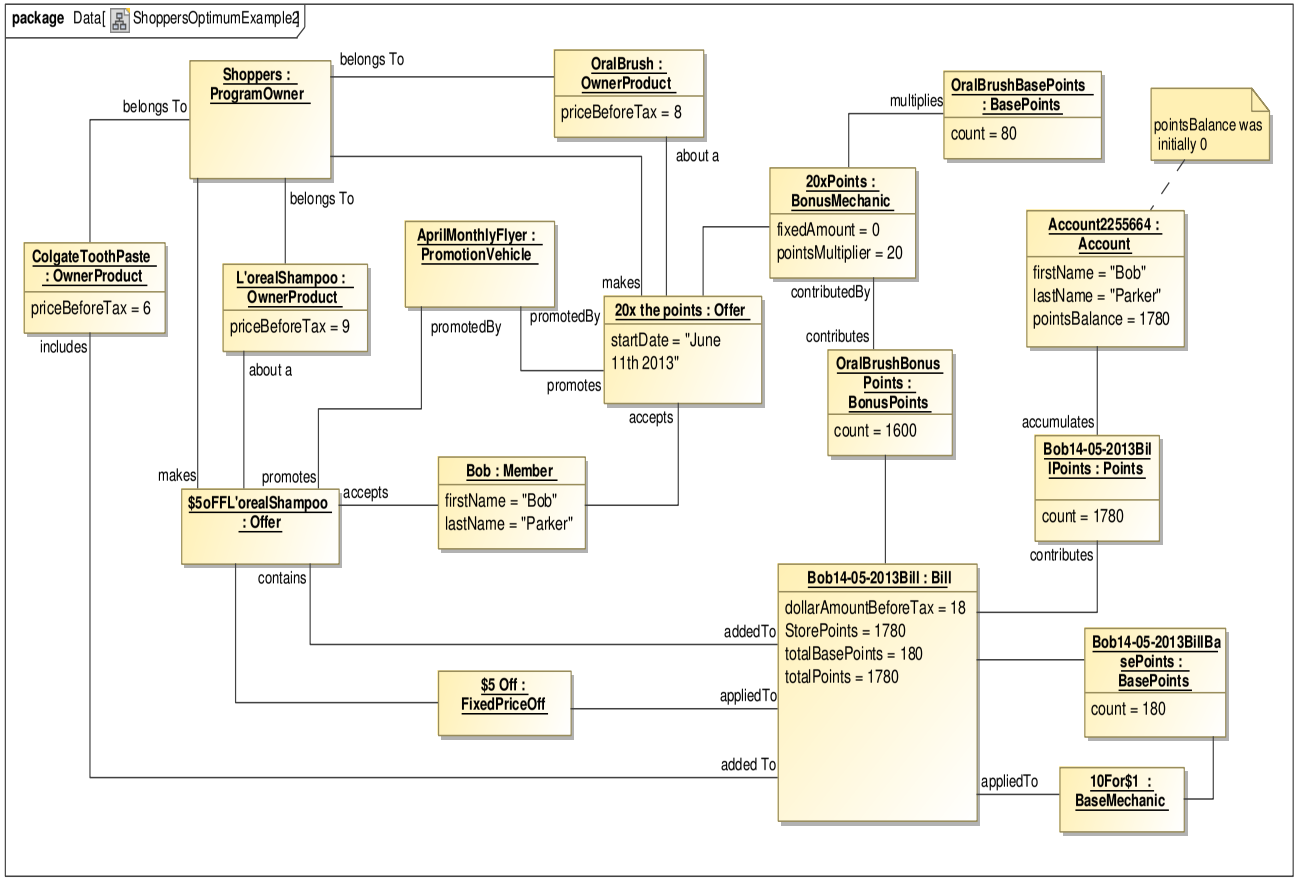


Figure B.3: Example 2 showing how a member earns bonus points through different bonus mechanics for two separate in store products, as well as a regular transaction with no offers for one product.

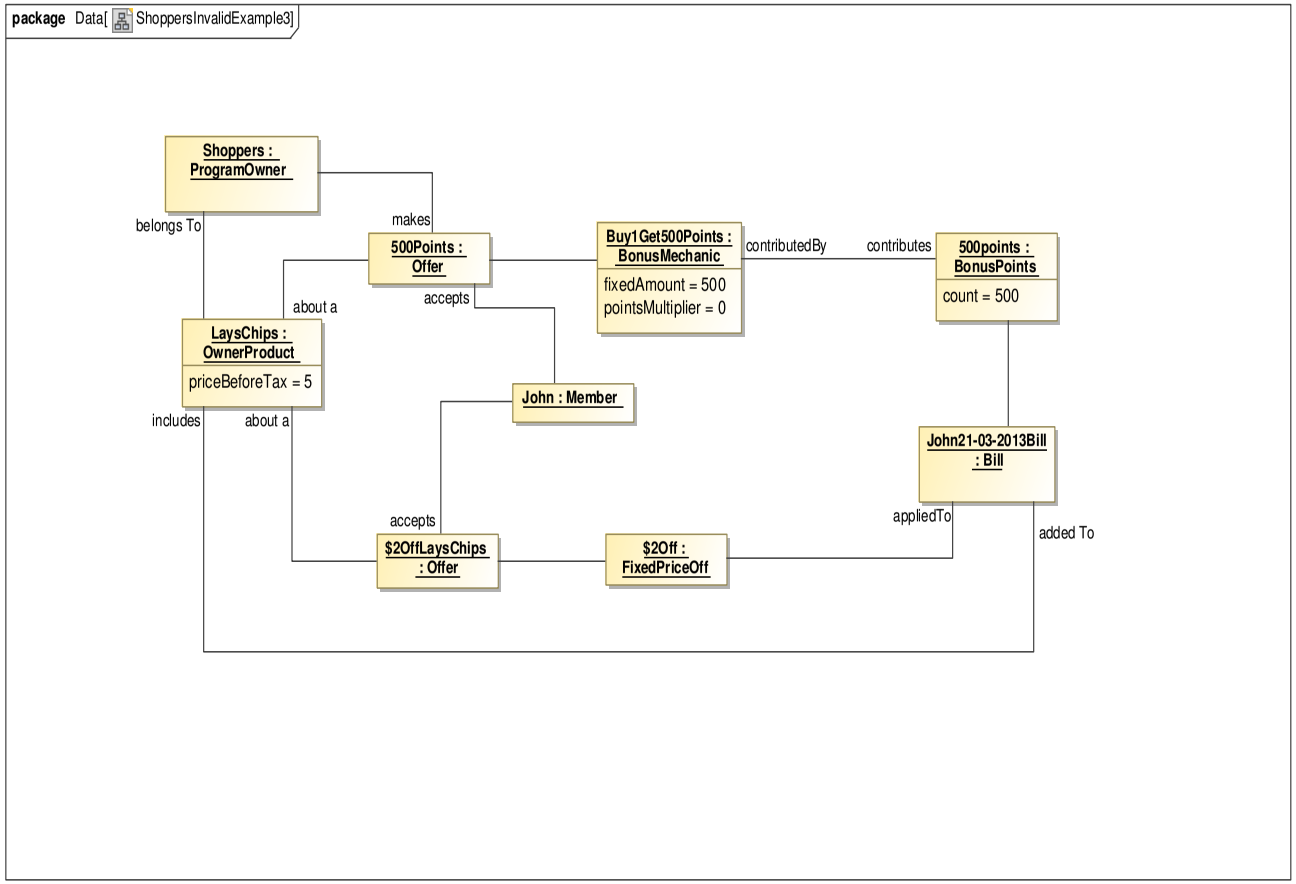


Figure B.4: Example 3 showing an invalid instantiation of the class diagram since it violates the following constraint: "At most one price or Bonus mechanic per offer".

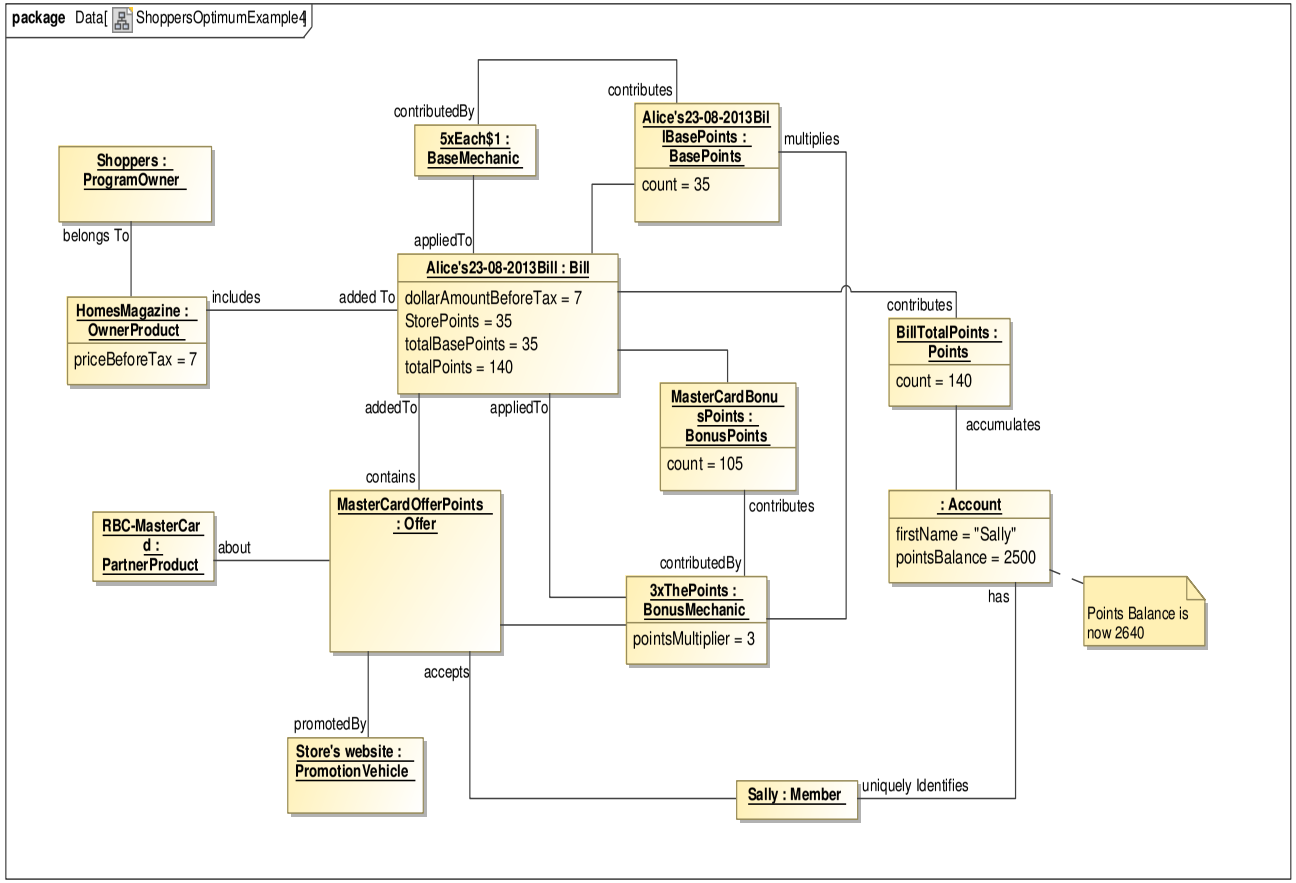


Figure B.5: Example 4 showing how a member earns bonus points for partner offers.

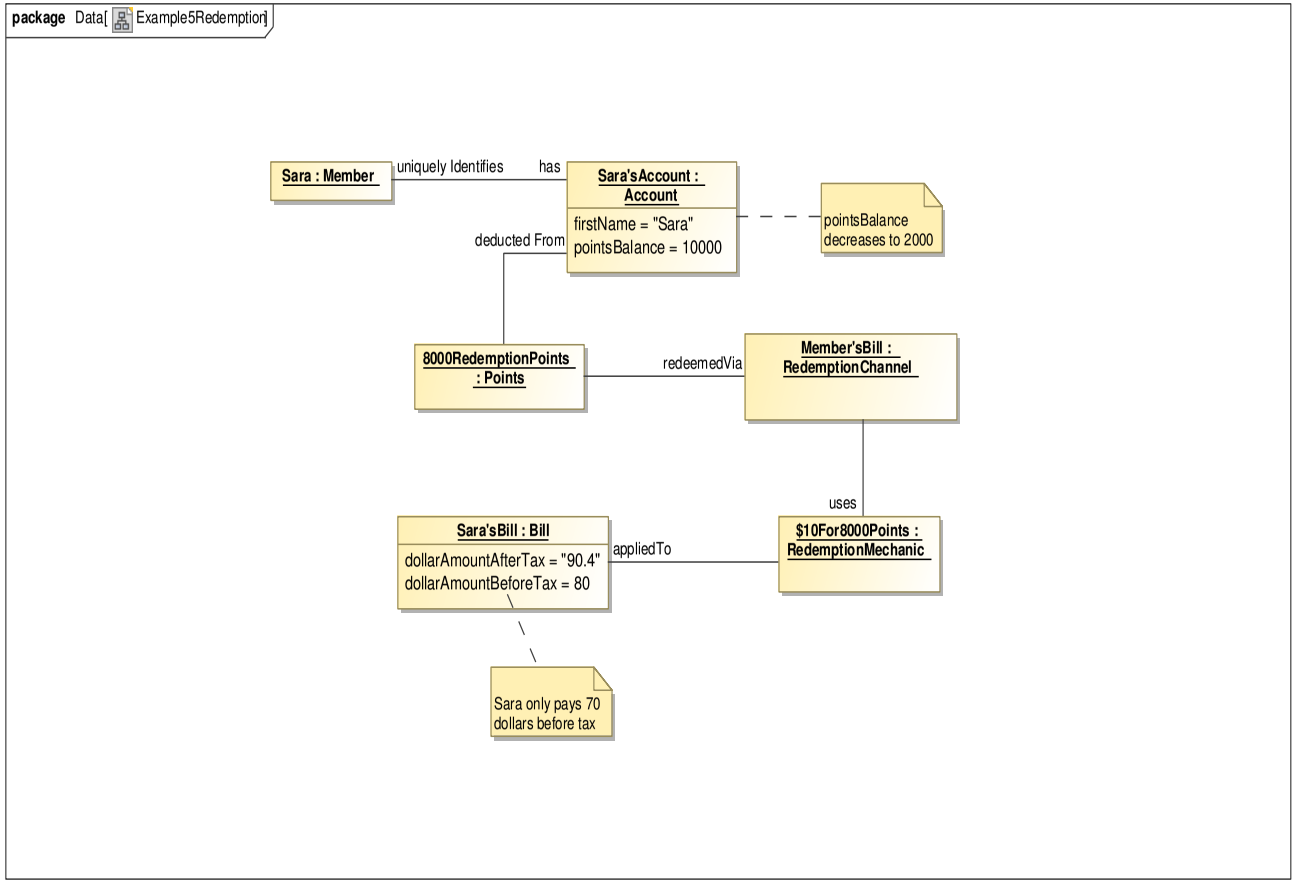


Figure B.6: Example 5 showing how a member can redeem some of her points.

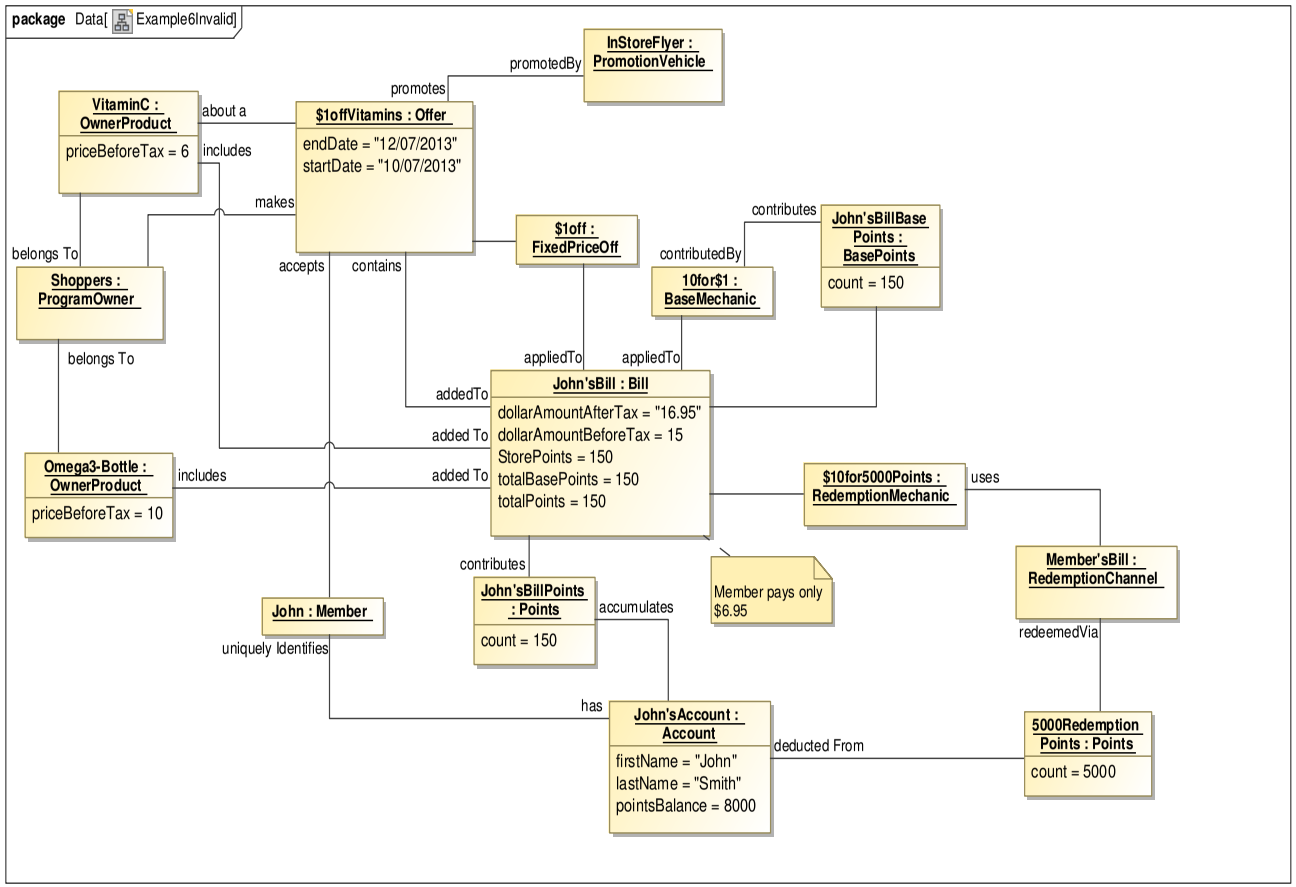


Figure B.7: Example 6 showing an invalid instantiation of the class diagram since it violates the following constraint: "You can't accumulate and redeem points for the same bill".

Appendix C

Participants' Solutions

C.1 Sample Object Diagram

We created a complete and correctly instantiated object diagram (Fig. C.1) that satisfies all experimental requirements to be used as reference when marking the participants' object diagrams. However, this is not the only complete and correct object diagram instance. There are other correct variations for this object diagram as discussed in Section C.2.1.

C.2 Detailed Analysis of Participants' Solutions

C.2.1 Correct Object Diagram Alternatives

In this section, we present other correct alternatives to our sample object diagram that we found in the participants' solutions.

Alternatives for *BonusMechanic* Class:
(1) A participant can include two instances of BonusMechanic as shown in Fig. C.1. However, some participants included one instance of the *BonusMechanic* class. For example, a participant can choose either (2) the *fixedAmount* attribute or (3) the *pointsMultiplier* attribute of the *BonusMechanic* class.

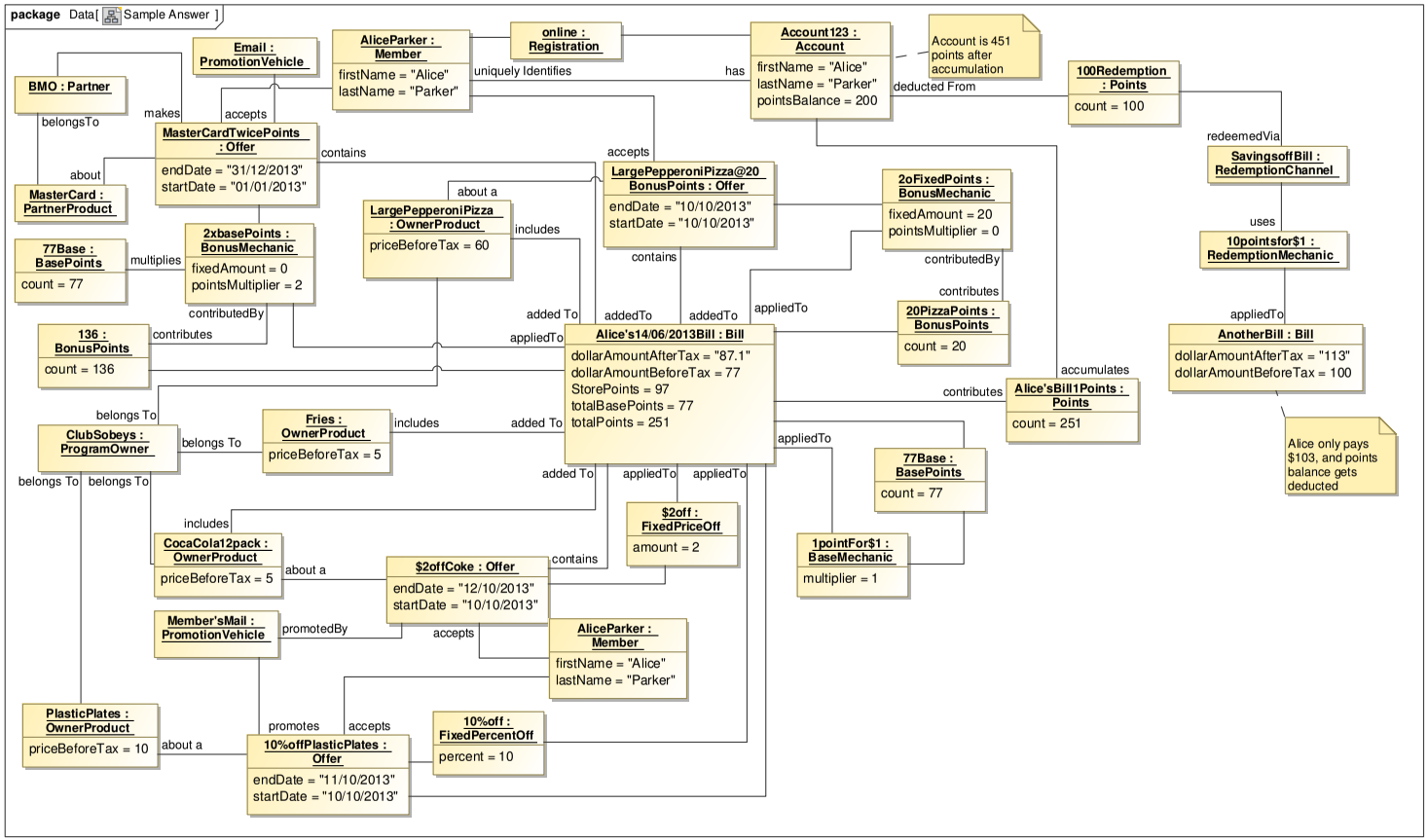


Figure C.1: Sample Object Diagram.

Alternatives for *RedemptionChannel* Class:
A participant can include one of two choices for a *RedemptionChannel*: (1) instant savings off a bill as shown in Fig. C.1, or (2) redeem their points through one of Club Sobeys partners such as Aeroplan.

Alternatives for *RedemptionMechanic* Class:
The redemption mechanic differs according to the redemption channel. For example, it could be (1) $1 for every 10 points if the *RedemptionChannel* is instant savings off a bill, or (2) 10 air miles for every 10 points if it is a conversion system between Sobeys and one of its partners such as Aeroplan.

Alternatives for *Registration* Class:
There are two possible alternatives for a member's registration to become a a member in Sobeys loyalty program: (1) either through the store's website (online registration) or (2) in person inside the store (in-store registration).

Alternative for *BaseMechanic* Class:
The participant can choose any value for the *BaseMechanic* class as long as it is consistent through the entire object diagram. For example, the mechanic could be 10 base points for every $1 instead of 1 point for every $1 (used in Fig. C.1).

Alternatives for *Partner* and *PartnerProduct* Classes:
(1) The participants can choose the *Partner* to be BMO bank. This is the partner suggested in the study materials. (2) Some participants chose the university as a partner. We think this choice is based on previous shopping experience. We considered the university to be a valid choice if the *PartnerProduct* was the student ID. This allowed us to make sure that there is logic and understanding behind the participant's choice and not just by chance. (3) The partner could also be any other bank than BMO. For example, some participants' chose CIBC bank. The corresponding partner product has to be changed accordingly. For example, if a participant chose CIBC as the partner then CIBC credit card or CIBC master card are valid partner products.

Alternatives for *PartnerOffer* Class:
(1) We designed the experiment so that partner offers use the *BonusMechanic* class to add more points. The points due to partner offers represent the difference between the *StorePoints* and *TotalPoints* attributes in the *Bill* class. However, some participants used

(2) the studentID card to be the partner product to get an offer of 10% off the entire bill instead of extra bonus points. We considered this choice correct as long as the mechanic used is applied to the entire member's bill. In that case, the values of *StorePoints* and *TotalPoints* attributes in the *Bill* class will be the same.

Alternatives for *PromotionVehicle* Class:
Each participant has to include two or more different promotion vehicles to promote offers according to the study requirements. Valid promotion vehicles for in-store offers include: (1) member's email, (2) member's mail, (3) in-store flyers, and (4) the store's website. Valid promotion vehicles for partner offers may also include: (5) the partner's website or (6) the partner's store/office.

Alternatives for *Account* Class:
(1) Updating the member's account *pointsBalance* after accumulation or redemption can be done either through an initial balance and then notes explaining the updates or (2) by simply including the final balance after accumulation and/or redemption have been applied.

C.2.2 Common Object Diagram Mistakes

In table C.1, we present common mistakes we found in the participants' object diagrams. We also present their relative frequency for the EDM group versus the control group.

C.2.3 Common Object Diagram Misses

In tables C.2 and C.3, we present common misses for classes and attributes in the participants' diagrams. We also present their relative frequency for the EDM group versus the control group.

C.2.4 Common Task 2 Mistakes

In table C.4, we present a summary of mistakes made by participants from both the EDM and control groups in Task 2 questions. We also present their relative frequency for the EDM group versus the control group.

Table C.1: Common mistakes in the participants' object diagrams.

Mistake	**Example**	**EDM Frequency**	**Control Frequency**	**Comments**
Using the product manufacturer as the program owner instead of Club Sobeys.	Kraft Foods is the program owner of mozarella cheese.	0	4	According to the study materials, the owner offering the rewards loyalty program is Club Sobeys and not each product manufacturer.
Using the partner as the partner product instead of the product name.	BMO bank is the partner product instead of BMO Mastercard.	1	2	The difference between a partner and partner product is mentioned in the study materials as a checkbox item.
Invalid owner products.	Mountain bike is not a valid grocery store product.	0	6	The study materials mentioned that Club Sobeys (program owner) is a grocery retailer and food distributor and hence a mountain bike wouldn't be a valid product to be sold in the store.
Applying base mechanic inconsistently.	Using 1 point for every $1 when calculating the total base points for the bill, while randomly choosing a count for base points per product.	0	4	There has to a be an explicit base mechanic for calculating the product base points which add to the total base points inside the Bill class according to the class diagram.
Choosing a partner.	Pepsico can not be a partner.	0	6	According to the class diagram, the partner product is different than the owner product. Hence, if a product sold in Sobeys, it can not qualify to be a partner product. In other words, the product manufacturer is not a valid partner. Also, an example for partner and partner product is mentioned in the study materials.
Applying a partner offer on an in-store product.	Pepsico offers the 12-pack pepsi cans at half price.	0	6	Depends on previous mistake.
Base Mechanic Definition	10x Base points	2	6	There has to be a clear definition of the mechanic. 10x is an ambigious base mechanic: 10 points for every $1 or every $10?
Base Points Calculation	Choosing a mechanic that awards 10 points for every $1 then adding 40 points for product priced at $2	0	7	

Mistake	**Example**	**EDM Frequency**	**Control Frequency**	**Comments**
Unclear Bonus Mechanic definition.	SobeysBonusMechanic	1	5	In order to judge the choice of bonus mechanic attributes, the grader must understand which bonus mechanic the participant intended to use.
Bonus Points Calculation	Choosing 10x base points as the bonus mechanic, but for a product that should have 10 base points, the participant has 200 bonus points.	0	7	Sometimes the bonus points calculation is wrong due to an incorrect value for base points.
Incorrect calculation of bill store points and total points.	(store points = total points) when having partner bonus points or difference between storePoints and totalPoints is the in-store bonus points instead of partner bonus points.	3	9	The formulas for calculating the storePoints and totalPoints are presented in the class diagram.
Using the redemptionChannel as the redemptionMechanic.	Using $10 for every 100 points as both the redemptionChannel and redemptionMechanic	1	3	The possible ways be which members can redeem their points is provided as a check box item in the study materials.
Invalid redemption channel.	"Sobeys Redemption channel" is an invalid redemption channel.	1	6	The possible ways be which members can redeem their points is provided as a check box item in the study materials.
Incorrect interpretation of registration.	Alice's registration instead of in-store or online registration.	2	11	Clarified by means of examples for the EDM group.
Applying accumulation and redemption for the same bill.	Earning and redeeming points for the same bill instance.	0	6	Violates a constraint in the class diagram.
Invalid redemption mechanics.	"Points Deduction" is not a valid redemption mechanic. The participants must specify what are the points redeemed for.	0	9	The possible ways be which members can redeem their points is provided as a check box item in the study materials.
Account accumulating base and bonus points directly.	Direct association between account and base/bonus points instead of through the bill instance.	0	6	The class diagram associations are clarified in terms of examples for the EDM group.

Mistake	**Example**	**EDM Frequency**	**Control Frequency**	**Comments**
Using the same instance of "points" as both accumulated and redeemed points.		0	5	Since the class diagram constraint mentions that you can't apply both accumulation and redemption for the same bill instance, then the points accumulated in a transaction can not be redeemed at the same time.
Having more than one partner product for the same bill.	BMO Mastercard and CIBC credit card adding bonus points to the same bill instance	0	2	Violates class diagram constraints that all offers made by a partner must be about one product.
Having the bill contribute base points instead of a "points" instance.		0	2	The bill collects base points, but contributes all the points per transaction. The associations are clarified by means of examples for the EDM group.
Creating an association between bonus mechanic and points instead of bonus mechanic and base points.		0	1	Violates the class diagram association that bonusMechanic multiplies basePoints.
Using base points as the redemption points deducted from account.		0	1	The class diagram distinguishes between baseMechanic and redemptionMechanic. Thus, base points can not be used as redemption points.
Creating an association between bonusPoints and redemptionMechanic.		0	1	The class diagram distinguishes between bonusMechanic and redemptionMechanic. Thus, bonus points can not be used as redemption points
Bill contributes redemption points instead of accumulation points.		0	1	The difference between accumulation and redemption is mentioned in the study materials. Further clarifications are given via examples to the EDM group.

Table C.2: Common missing objects in the participants' object diagrams.

Missed Object	**EDM Frequency**	**Control Frequency**	**Comments**
OwnerProduct	0	5	
Partner	2	2	The EDM participants who were missing the partner object got the partnerProduct correct, while the control participants were missing the partnerProduct as well. This suggests that the missing Partner object for the control participants was probably due to lack of understanding, while it may not be the case for the EDM participants.
PartnerProduct	0	2	The control participants who missed the Partner object missed the PartnerProduct object.
PartnerOffer	0	3	The two control participants who missed the Partner and partnerProduct objects missed the PartnerOffer object.
Missing a second PromotionVehicle (requirement).	2	7	The participants might have missed this requirement from the study materials, or they didn't understand that the promotion vehicle represented the "sales channels" as mentioned in the study materials.
FixedPriceOff	2	4	
FixedPercentOff	5	7	Participants from both groups missed this object because they used the FixedPriceOff. They solved the question related to FixedPercentOff in Task 2 correctly, which suggests that missing this object was not due to lack of understanding.
BaseMechanic	0	1	
BasePoints	0	1	
BonusMechanic	0	4	
BonusPoints	0	1	
Card	3	1	
Points (Accumulated)	2	5	
Points (Redeemed)	1	6	
RedemptionChannel	0	2	
RedemptionMechanic	0	3	
OwnerProduct that has no offers (requirement)	2	8	

Table C.3: Common missing attributes in the participants' object diagrams.

Missed Attributes	EDM Frequency	Control Frequency	Comments
startDate, endDate in Offer	3	7	
amount in FixedPriceOff	2	3	
amount in FixedPercentOff	2	2	Participants from both groups included the percentage in the name of the object, but didn't include it as an attribute.
No attributes in BonusMechanic	1	4	The EDM participant and 1 control participant included the fixed amount of bonusMechanic in the name of the object, but not as an attribute.
dollarAmountAfterTax attribute in Bill	2	4	

C.2.5 Domain Questions Raised by Participants

In Table C.5, we present the domain questions raised by the participants during the experimental sessions. We also present their relative frequency for the EDM group versus the control group.

C.2.6 Commentary on the Correlations between the Participants' Answers

Most of the control group mistakes and/or misses in their object diagrams were related to the following classes: (1) RedemptionMechanic, (2) RedemptionChannel, (3) BonusMechanic, (4) Partner, (5) PartnerOffer, (6) Registration, (7) PromotionVehicle, (8) OwnerProduct, (9) calculation of Bill attributes, (10) fixedPercentOff mechanic, and (11) the associations connected to the "points" class. Interestingly, when we asked the control participants to mention which classes were the hardest to comprehend during their feedback, they mentioned only four of these classes: BonusMechanic, RedemptionMechanic, PartnerOffer, and Bill. Our observations during feedback is that the control participants had problems with understanding redemption in general which they only expressed in terms of "RedemptionMechanic". The EDM participants, on the other hand, had no problems with redemption given that they mentioned that Example 5 that explained redemption

Table C.4: Common mistakes in the participants' task 2 solutions.

Task 2	**EDM Frequency**	**Control Frequency**	**Comments**
Question 1	1	8	
Question 2	3	7	
Question 3	2	3	
Question 4	5	6	
Question 5	0	7	Three participants scored zero, and four answered with either base or bonus points instead of both.
Question 6	0	2	
Question 7	4	9	
Question 8	0	8	
Question 10	0	6	One participant scored zero and five participants scored only 1 point for describing either the fixedAmount or the pointsMultiplier attributes correctly.
Question 11	1	4	One EDM participant got only 1 mark for answering with BaseMechanic only. One control participant scored zero while the other three got only 1 mark for answering with either baseMechanic or bonusMechanic instead of both.
Question 12	1	4	
Question 13	4	9	One EDM participant scored zero while two others got 1 mark for describing only storePoints correctly. Five control participants scored zero while the other 4 got got 1 mark for describing only storePoints correctly.
Question 14	3	4	
Question 15	2	12	The two EDM participants got 4 out 5 marks for missing the accumulated "points" instance. These were the two participants who missed this object from the object diagram. The answers for the control group varied greatly.

Table C.5: Domain questions raised by the control and EDM participants.

Question	**EDM Frequency**	**Control Frequency**
What do you mean by promotion vehicle?	0	8
What's the difference between program owner and partner?	0	7
What's the difference between store points and total points attributes in the bill class?	2	9
Why do we need owner product and partner product?	0	6
Can the same partner be used for both accumulation and redemption?	0	4
What's the difference between the associations associated with the points class?	0	10
Why do we need "points" in addition to base and bonus points?	0	8
What's the difference between redemption mechanic and redemption channel?	1	11
Can I include more than one bill instance because that would be the only way to satisfy some constraints?	3	0
Do I have to use the same base mechanic for all products?	0	2
Can I instantiate different products for the different mechanics?	2	3
I don't understand why is there a separate association between partner product and offer, and owner product and offer?	1	10
I can't include two different price mechanics in my object diagram, right?	0	5
Why would bill collect points?It should only contribute.	0	6
Can you help me with the difference between base and bonus mechanics? I think we only need bonus mechanic for extra points.	0	7

was clear enough.

These results also conform with the participants' Task 2 solutions. For example, all control participants answered the question in Task 2 related to "FixedPercentOff" mechanic correctly and they didn't report any problems understanding it in the de-briefing questionnaire, which suggests that they understood what the mechanic meant (probably due to previous shopping experience). However, they asked whether they could include more than one price mechanic in their object diagram, which suggests that they might not have understood the constraint associated with the price mechanics correctly. The constraint mentioned that one can only have one priceMechanic per product, but this does not mean that one can not have multiple products with different offers. Although the EDM participants did not specifically mention Example 3 (related to this constraint) as being one of the most useful examples, it's likely that the constraint was clearly conveyed in the example that they didn't think it was of a high difficulty level to mention.

For the following classes and/or associations: RedemptionMechanic, RedemptionChannel, BonusMechanic, Bill, Points, all control participants mentioned that they were difficult to understand in the de-briefing questionnaire. They also asked clarifying questions about them during the experimental session when creating their object diagrams. The EDM group mistakes for these classes were always far less than half of the mistakes made by the control group. Questions 5, 8, and 10 in Task 2 were related to these classes. All EDM participants answered them correctly, while less than half of the control group were able to solve them.

For the following classes and/or associations: Partner, PartnerOffer, Registration, PromotionVehicle, OwnerProduct, the control group participants answered the Task 2 questions related to them incorrectly and also raised some domain questions related to them during the experimental session. However, they didn't mention them in the de-briefing questionnaire. This suggests that the participants might have mistakenly thought that they understood them after finishing the experimental tasks and that's why they did not mention them as difficult parts in the de-briefing questionnaire (except for only one participant who said partner and programOwner classes were hard to understand).

References

[1] Magicdraw. http://www.nomagic.com/products/magicdraw.html. Accessed: 2013-09-11.

[2] The R project for statitical computing. http://www.r-project.org/. Accessed: 2013-09-11.

[3] Michał Antkiewicz, Kacper Bak, Dina Zayan, Krzysztof Czarnecki, Andrzej Wasowski, and Zinovy Diskin. Example-driven modeling using Clafer. In *First International Workshop on Model-driven Engineering By Example*, 2013.

[4] Kacper Bak, Zinovy Diskin, Michał Antkiewicz, Krzysztof Czarnecki, and Andrzej Wasowski. Partial instances via subclassing. In *6th International Conference on Software Language Engineering*, 2013.

[5] Kacper Bak, Krzysztof Czarnecki, and Andrzej Wasowski. Feature and meta-models in clafer: Mixed, specialized and coupled. In *International Conference on Software Language Engineering*, pages 291–301, 2010.

[6] Kacper Bak, Zinovy Diskin, Michał Antkiewicz, Krzysztof Czarnecki, and Andrzej Wasowski. Clafer: Unifying class and feature modeling. *Journal paper. Submitted for review.*, 2013.

[7] Kacper Bak, Dina Zayan, Krzysztof Czarnecki, Michał Antkiewicz, Zinovy Diskin, Andrzej Wasowski, and Derek Rayside. Example-driven modeling. Model = Abstractions + Examples. In *New Ideas and Emerging Results (NIER) track of ICSE'13*, 2013.

[8] Narciso Cerpa and June M. Verner. Why did your project fail? *Commun. ACM*, 52(12):130–134, December 2009.

[9] Michelene T.H. Chi, Paul J. Feltovich, and Robert Glaser. Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5(2):121 – 152, 1981.

[10] Hyun Cho, J. Gray, and E. Syriani. Creating visual domain-specific modeling languages from end-user demonstration. In *Modeling in Software Engineering (MISE), 2012 ICSE Workshop on*, pages 22–28, 2012.

[11] R.A. DeMillo, R.J. Lipton, and F.G. Sayward. Hints on test data selection: Help for the practicing programmer. *Computer*, 11(4):34–41, 1978.

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

[13] Krzysztof Czarnecki Dina Zayan, Michal Antkiewicz. Effects of using examples on structural model comprehension: A controlled experiment, 2014.

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

[15] Mary L Gick and Keith J Holyoak. Schema induction and analogical transfer. *Cognitive psychology*, 15(1):1–38, 1983.

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

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

[18] John Hutchinson, Jon Whittle, Mark Rouncefield, and Steinar 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] Natalia Juristo and Ana M. Moreno. *Basics of Software Engineering Experimentation*. Springer Publishing Company, Incorporated, 1st edition, 2010.

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

[22] B.A. Kitchenham, S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. 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] Soren Lauesen and Otto Vinter. Preventing requirement defects: An experiment in process improvement. *Requirements Engineering*, 6(1):37–50, 2001.

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

[25] Lucy Mendel, Daniel N. Jackson, Arthur C. Smith, and Lucy Mendel. Modeling by example, 2007.

[26] Ryszard S. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20(2):111 – 161, 1983.

[27] S. H. Muggleton. Inductive logic programming, 1991.

[28] Dan North. Introducing BDD. *Better Software*, 12, 2006.

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

[30] Jeff Offutt and Aynur Abdurazik. Generating tests from UML specifications. In Robert France and Bernhard Rumpe, editors, *UML'99—The Unified Modeling Language*, volume 1723 of *Lecture Notes in Computer Science*, pages 416–429. 1999.

[31] I. Olkin. Contributions to probability and statistics:essays in honor of harold hotelling. *Stanford University Press*, 1960.

[32] Rajesh Parekh and Vasant Honavar. Grammar inference, automata induction, and language acquisition. *Handbook of natural language processing*, pages 727–764, 2000.

[33] Shelly S. Park. Communicating Domain Knowledge through Example-Driven Story Testing. Master's thesis, University of Calgary, Alberta, 2011.

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

[35] Peter Pirolli. Effects of examples and their explanations in a lesson n recursion: A production system analysis. *Cognition and Instruction*, 8(3):207–259, 1991.

[36] Helen C. Purchase, Linda Colpoys, Matthew McGill, David Carrington, and Carol 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.

[37] F. Ricca, M. Di Penta, Marco 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.

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

[39] G. Scanniello, F. Ricca, and Marco 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.

[40] Silke Schworm and Alexander Renkl. Learning by solved example problems: Instructional explanations reduce self-explanation activity. In *IN*, pages 816–821. Erlbaum, 2002.

[41] Bonita Sharif and Jonathan I Maletic. An empirical study on the comprehension of stereotyped uml class diagram layouts. In *Program Comprehension, 2009. ICPC'09. IEEE 17th International Conference on*, pages 268–272. IEEE, 2009.

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

[43] Gregory Trafton, Brian J. Reiser, and Greg Trafton. Studying examples and solving problems: Contributions to skill acquisition.

[44] Tamara van Gog, Liesbeth Kester, and Fred Paas. Effects of worked examples, example-problem, and problem-example pairs on novices' learning. *Contemporary Educational Psychology*, 36(3):212 – 218, 2011.

[45] Rini van Solingen, Vic Basili, Gianluigi Caldiera, and H. Dieter Rombach. *Goal Question Metric (GQM) Approach*. John Wiley & Sons, Inc., 2002.

[46] Jos Warmer and Anneke Kleppe. *The Object Constraint Language: Getting Your Models Ready for MDA*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2 edition, 2003.

[47] Bernard L Welch. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

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

[49] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Bjöorn Regnell, and Anders Wesslén. *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[50] Shehnaaz Yusuf, Huzefa Kagdi, and Jonathan 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.