An Empirical Evaluation of Visualisation in Software Design Modelling: the VCL vs UML+OCL experiment

Nuno Amálio
Birmingham City University
School of Computing and Digital Technology
Millennium Point, Cursor Street
Birmingham B6 7XG, UK

Lionel Briand
SnT Center
University of Luxembourg
4, rue Alphonse Weicker
L-2721 Luxembourg

Pierre Kelsen
University of Luxembourg
6, rue R. Coudenhove-Kalergi
L-1359 Luxembourg

TR-LASSY-13-05

Abstract

Popular notations of software and systems engineering, such as the standards UML and SysML, have an intrinsic visual nature, enabling them to tap into the cognitive benefits that visual representations are know to provide. However, these widespread notations fail to exploit the full cognitive potential of visual representations; despite their visual appeal, they fail to be cognitive effective. The Visual Contract Language (VCL) tries to improve the visual effectiveness and level of rigour of popular visual notations of software and systems design; it tries to fully exploit the cognitive potential of visual representations and increase the range of things that can be described visually. VCL embodies a diagrammatic modelling approach that does pictorially what is traditionally done textually.

This report presents the work that was undertaken to rigorously evaluate VCL. It presents the design of a controlled experiment to evaluate the effectiveness of VCL together with the results of the experiment. The experiment compares VCL against UML supplemented with OCL, which are widely seen as industry standards.

Contents

List of Figures

List of Tables

Acronyms

Assertion Diagram (AD)

A VCL diagram that describes predicates (or assertions).

Behaviour Diagram (BD)

A VCL diagram that defines a map over the behavioural units of a system or a package.

Confidence Interval (CI)

Indicates the precision of a point estimate or measurement, providing a range of plausible values within which we can have a degree of confidence that the estimate is not due to chance.

Contract Diagram (CD)

A VCL diagram that describes contracts made-up of a pre- and a post-condition.

Effect Size (ES)

Quantitative estimate of the magnitude of some effect of interest.

Null Hypothesis Significance Testing (NHST)

A statistical method to test research hypotheses based on stating a null and an alternative hypothesis regarding differences or effects that occur in the population. The null hypothesis represents the "no difference" or "no effect" and the alternative hypothesis represents the occurrence or presence of the difference or effect. The hypothesis is tested using statistical approaches by rejecting the null-hypothesis to calculate a probability known as the p-value..

Object Constraint Language (OCL)

Unified Modelling Language (UML)'s satellite textual language for expressing constraints of invariants and operations..

Package Diagram (PD)

A VCL diagram that defines packages, specifying the relation of the package being defined with other packages. Packages act as stand-alone containers or modules containing structures and behaviour.

Structural Diagram (SD)

A VCL diagram that defines the structures that together makes the state space of a package or a system, and identifies the invariants that constrain the state space.

Systems Modelling Language (SysML)

A general-purpose notation for systems engineering that builds up on the UML.

Unified Modelling Language (UML)

A mainstream visual modelling language for modelling software designs.

Visual Contract Builder (VCB)

The tool that supports VCL.

Visual Contract Language (VCL)

A visual and formal modelling language that aims to improve mainstream modelling notations, trying to do diagrammatically what UML does textually.

Chapter 1

Introduction

There is an increasing push for modelling in software and systems engineering. This necessity does not emerge any longer from guidelines of good engineering practice; instead, it emerges from the deep bottom of the practice of software and systems engineering. Modelling is simply the natural response to a need for abstraction, and better means for tackling complexity and the development of large systems.

Popular notations of software and systems engineering, such as the standards UML and Systems Modelling Language (SysML), have an intrinsic visual nature, which enables them to tap on the cognitive benefits that visual representations are know to provide. However, these widespread notations fail to exploit the full cognitive potential of visual representations; despite their visual appeal, they fail to be cognitive effective.

The Visual Contract Language (VCL) [AG15, AK10, AKMG10, AGK11] tries to improve the visual effectiveness and level of rigour of popular visual notation of software and systems design. VCL tries to fully exploit cognitive potential of visual representations and tries to increase the range of things that can be described visually; it proposes an approach to describe diagrammatically what is traditionally described textually.

This document describes the design of a controlled experiment to evaluate the effectiveness of VCL. This is done by comparing VCL against UML supplemented with the Object Constraint Language (OCL), which are widely seen as industry standards. The following sections briefly explain the motivations behind the visual nature of VCL, give a short explanation of VCL and its development and present the actual experiment.

1.1 Why VCL?

When solving problems, human beings use both internal representations, stored in their brains, and external representations, recorded on a paper or some other medium [LS87]. It is today quite widely acknowledged, in the field of cognitive science, that the form of external representations matters to the agents' performances in achieving various cognitive tasks, such as problem-solving or decision-making [LS87, Zha97]. This is because the form of a representation determines what information can be perceived, what cognitive processes can be activated, and what structures can be discovered from the specific representation. According to Zhang [Zha97], "external representations are not simply inputs and stimuli to the internal mind; rather, they are so intrinsic to many cognitive tasks that they guide, constrain and even determine cognitive behaviour".

Humans like to process information in pictorial form [Pin90, LS87, Goo00]. This is because diagrams facilitate easy perceptual inferences [LS87]. They tap into the capabilities of the power-

ful and highly parallel human visual system, constituting an effective method of communication. The effectiveness of visual descriptions is evidenced by its relevance to enginering [Fer77, Fer92]. Visual thinking is seen as an integral part of engineering; thinking, designing and communicating with pictures are recognised as essential engineering activities. Like in traditional engineering, diagrammatic representations have also become an integral part of the language of software engineering [Moo09]. Diagrams are considered to be a language of IT practice. It is, therefore, not a surprise that software engineering visual languages have been advocated for decades with the purpose to facilitate human communication and problem solving [Har88]. Popular engineering visual languages include Harel's statecharts [Har87] and the UML in software engineering, and the SysML [OMG12] in systems engineering.

1.2 The Visual Contract Language

VCL [AG15, AK10, AKMG10, AGK11] is designed to specify models of software designs visually and formally. It is motivated by: (a) the lack of rigour and formality in the semantics definition of mainstream visual languages, such as the UML, which precludes means of formal verification and validation, (b) the fact that not all properties can be described visually in mainstream visual languages, such as the UML (UML compensates this with the satellite textual OCL, which describes invariants and operations), and (c) the fact that mainstream visual languages, such as the UML, lack cognitive effectiveness [Moo09, MvH08].

VCL's novel features lie in its capacity to describe predicates visually and in its visual approach to behavioural modelling based on *design-by-contract* (pre- and post- conditions). Other novel feature of VCL lies in its symmetric packaging mechanisms for coarse-grained separation of concerns that is inspired in the ideas of aspect-orientation. VCL has a type system [Amá12a], a formal semantics and a tool [AGK11][1], theVisual Contract Builder (VCB). VCB implements VCL's type system for checking the consistency of VCL models and produces Z specifications from VCL models.

VCL has been, since its inception, applied to real-world problems. These applications resulted in improvements to the language. The case studies to which VCL has been applied constitute challenges posed by research communities; VCL has not, however, been applied yet in an industrial setting. The main VCL case studies are summarised in table 1.1. Two masters thesis of the University of Luxembourg have been devoted to VCL [Lee11, Tob12].

In [TRA12a, Tob12, TRA12b], VCL is compared with other visual modelling languages, that, like VCL, are capable of expressing predicates visually. In this study, VCL outperformed the other two competitors mainly due to its superior tool support.

Due to the several iterations of application and improvement to which VCL has been subject to, in particular the applications in [AKMG10, Amá11, LA12b, LA12a, Amá12b, TRA12b], we now feel that VCL and VCB are reaching stability. This is confirmed by the recent marked project on VCL in the context of the masters course *Model Driven Software Development* at the University of Luxembourg. We now feel that it is the right time to do a controlled experiment on VCL. A brief introduction to VCL is given in appendix A.

1.3 Related Work

The existing literature includes a number of studies on data model understanding. The findings of a number of relevant studies are summarised below.

[1] `http://vcl.gforge.uni.lu`

Table 1.1 Main case studies to which VCL has been applied.

Case Study	Description
Crisis management system [AKMG10]	The first large case study to which VCL was applied. A challenge for aspect-oriented modelling approaches to evaluate how they tackle large and complex case studies. This resulted in the mechanism of packages and associated decomposition and composition mechanisms. The model for this case study was developed when tool support for VCL was not yet available.
Cardiac Pacemaker [LA12b, LA12a]	This is a benchmark cases study: a challenge coming from the formal methods community. It was developed using VCL's tool, and most work was done in the context of a masters thesis.
Barbados Crisis Management System [Amá12b]	This is a smaller and more focused version of the crisis management system (developed in [AKMG10]), which was developed in response to a call for comparing modelling approaches.
Secure Simple Bank [Amá11]	VCL's toy case study to experiment with all of VCL's features.

1.3.1 Data Modelling

Both VCL and UML take an approach to modelling that is based on the idea of a data or conceptual model. There are a number of studies that investigated the effectiveness of data modelling methods and notations:

- In [KM95], Kim and March compare the effects of the *extended entity-relationship* notation (EER), an *entity-attribute-relationship* approach against NIAM (Nijssen information analysis methodology), an *object relationship* approach. The study examined the effects of the investigated formalisms on modeller performance in developing a data model and on user performance in validating a data model. The study uses as dependent variables the task performance on modelling and comprehension and the perceived usefulness of the formalisms. The study did not encounter significant differences in user comprehension, but they found that EER outperformed NIAM in the model development tasks; EER was perceived as more useful than NIAM by modellers.

- Moody [Moo02]investigates user comprehension of large data models using different data modelling approaches. This study uses as dependent variables both comprehension and validation speed, accuracy, and efficacy. Moody's study found that his method for managing the complexity of data models (the levelled data models approach) improves end-user understanding of data models.

1.3.2 UML

- Purchase et al [PCM$^+$01, PCMC02] investigates the comprehension of different variants of UML class and collaboration diagrams. For class diagrams, the experimental task consisted of matching a textual specification against a set of diagrams, indicating whether each diagram matches the specification or not. For collaboration diagrams, the task consisted of matching the diagram against a textual description of the diagram in pseudo-code. For

class diagrams, the study concluded that there is no best syntactical variant; that the best one depends on the task in which it is used.

- Briand et al [BLPYB05] evaluates the effectiveness of model development using UML with and without OCL. This is done with the aim of evaluating the benefits of OCL in development activities associated with design models. The evaluation is based on three software engineering tasks: detection of model defects through inspections, comprehension of the system logic and functionality, and impact analysis of changes. Results show that significant benefits can be obtained by using UML together with OCL, but this requires thorough training of participants.

- Arisholm et al [ABHL06] studies the impact of UML-based software documentation on software maintenance. Subjects then performed code modifications with and without the UML. Results show that, for complex tasks and past a certain learning curve, the availability of UML documentation may result in significant improvements in the function correctness of changes as well as the quality of their design; however, usage of UML does not seem to provide any savings of time.

Chapter 2

The Scrutiny and its Experiment

This chapter defines the scope of the scrutiny presented here and lays down the foundations of its underlying controlled experiment.

2.1 Scope, Objective and Effects

The experiment presented here aims to investigate the effectiveness of VCL as a visual notation. We want to know whether VCL's novel visualisations provide any modelling or design advantage over existing standard notations. This is done by comparing VCL against the UML standard and its satellite textual OCL notation. The experiment's comparison focuses on design models made of a state definition, state invariants and state dynamics (behaviour or operations). The evaluation is to focus on the expression of invariants and operations to better contrast the visual (VCL) against the textual (OCL).

The experiment's objective is as follows:

> Evaluate the effectiveness of VCL on user performance in a set of tasks associated with constructing and using design models by comparing VCL against UML and its satellite textual language OCL.

The experiment considers two different modelling perspectives: *modellers* and *model end-users*. It investigates the following aspects:

1. **Model construction.** Software engineers construct models to describe system or component designs. We assess their performances and the the way they perceived their modelling experience with the languages under scrutiny.

2. **Problem comprehension.** It is known that modelling leads to a better comprehension of the modelled problem. Hence, we compare the languages under study to assess performances with respect to problem comprehension from the modellers' perspective.

3. **Model Usage.** Once produced, models are used in a various settings by different stakeholders. We evaluate the performance of end-users in tasks related to the usage of models.

4. **Usefulness, Ease of Use, Usability and Overall Appraisal.** We evaluate the perceptions of subjects by taking into account their experience in carrying out the experiment's tasks.

2.1.1 What this experiment is not about

VCL includes other visual artifacts that are not investigated in this experiment, namely, VCL's visual approach to coarse-grained separation of concerns [AKMG10]. The evaluation of these visual constructions is left for future work.

2.2 Research Questions

Based on the experiment's objective, we formulate the following research questions:

- *RQ1: Is the performance of modellers in building software designs better with VCL than UML+OCL?*
- *RQ2: Is modellers' performance in problem comprehension accrued from modelling better with VCL than UML+OCL?*
- *RQ3: Is the performance of end-users in tasks related to the usage of software design models better with VCL than UML+OCL?*
- *RQ4: Is VCL perceived as being more useful and easy to use than UML+OCL?*
- *RQ5: Is VCL's usability better than UML+OCL's?*
- *RQ6: How is the overall perception of VCL in comparison to UML+OCL?*

RQ1 and RQ2 evaluate the effectiveness of the notation from the perspective of modellers. RQ3, on the other hand, evaluates the effectiveness of the notation from the perspective of model users. RQ4 assesses the perceived usefulness and ease of use of both notations. RQ5 assesses VCL's usability in comparison to UML+OCL. Finally, RQ6 tries to infer the overall perceptions of the languages under scrutiny at the light of a comparison, based on the experience of subjects in using both notations.

2.2.1 RQ1: Model Construction

RQ1 is evaluated using a model construction task that starts from a case study narrative. To facilitate the analysis, the modelling is separated into pertinent aspects of design modelling, namely: a definition of state structures (the state space), a definition of constraints over the state space (invariants) and definitions of behaviour as pre- and post-conditions (operation contracts).

We separate the pertinent aspects of design models into *structure* and *behaviour*. We consider that structure is itself divided into: a definition of the structures that make a design (such as a UML class diagram), and its associated constraints (invariants); we say that the underlying structures or abstractions together with its constraints define the state space of a system (all possible allowed states). Behaviour is described as operation contracts following the *design by contract* paradigm [Mey92].

We compare the following: (a) VCL structural diagrams against UML class diagrams, (b) VCL assertion diagrams of invariants against OCL constraints of invariants, and (c) VCL assertion and contract diagrams of operations against OCL constraints of operations. The comparison is based on the following criteria:

- *Completeness.* This measures how much is modelled of the requirements. We partition the requirements and then rank their satisfaction in the model according to an ordinal scale from 4 (total satisfaction) to 0 (no satisfaction).

- *Accuracy.* This measures the quality of the produced model against test cases. The satisfaction of the test cases, checked manually by experts, is based on an ordinal scale from 4 (total satisfaction) to 0 (no satisfaction).

2.2.2 RQ2: Problem Comprehension

The understanding gained from modelling[1] is evaluated with a questionnaire, comprising a set of multiple choice questions. This aims to evaluate the effect of modelling using the languages under study in the modellers' understanding of the problem.

2.2.3 RQ3: Model Usage

This is evaluated with the following tasks:

- *Defect detection.* This is related to the common task of model inspection. It consists of identifying defects in a given model with seeded errors. We measure the proportion of encountered defects.
- *Model Comprehension*, or how well users understand of models, is assessed through a questionnaire containing multiple choice questions about a given design. We measure the proportion of correct questions. This is similar to problem comprehension, but more focussed on model usage.

2.2.4 RQ4, Usefulness and Ease of Use

The notations under study are assessed at the light of *perceived usefulness* (PU) and *perceived ease of use* (PEoU) [Dav89]. PU is the degree to which someone believes that using a particular system would enhance their job performance. PEoU is the degree to which someone believes that using a particular system or technology would be free of effort. This is assessed through the debriefing survey.

2.2.5 RQ5, Usability

We study usability using criteria related to modelling notations, namely: readability, navigation, maps and overviews, live error checking, look and feel, learnability and comfort/satisfaction. This is assessed through the debriefing survey.

2.2.6 RQ6, Overall Perception

We assess the perceptions of subjects with respect to the languages under scrutiny based on their experiment experience. This is assessed through the debriefing survey.

2.3 Variables and Hypotheses

The experiment has one independent variable or factor, referring to the notation, with two treatments, *VCL* and *UML*, corresponding to the use of VCL or UML+OCL, respectively. The dependent variables are listed in table 2.1, which indicates the variable's category, research

[1] It is a known fact of the requirements engineering and software modelling literature that the process of modelling aids in gaining a better understanding of the problem. The aim is to investigate whether VCL's visualisations are effective in aiding reasoning and in producing useful inferences.

question and type according to information that it holds. The dependent variables are named according to the following convention: abbreviation of category (e.g. Co = completeness; Ac = accuracy) followed by abbreviation of measured modelling aspect — either state space (S), invariants (I) or operations (O); *CoS* stands for completeness of state space. We use two types of variables: (i) *proportion* is obtained by dividing the obtained quantity by the maximum quantity, yielding a continuous number between 0 and 1, and (ii) *nominal* or *categorical* whose possible values are drawn from a finite and discrete set.

The experiment's hypotheses are listed in table 2.2. They are formulated from the dependent variables and the experiment's two treatments. For each dependent variable, we formulate the null hypothesis H_i^0 (no difference between the notations), and the alternative hypothesis H_i^a (VCL outperforms UML+OCL).

2.4 Recruitment and Experiment Venues

The experiment was carried out in four venues: (a) Faculty of Sciences of the University of Lisbon, Portugal (FCUL); (b) Faculty of Science and Technology of the New University of Lisbon, Portugal (FCT/UNL); (c) University of Luxembourg (UL); and (d) University of York, UK (UY). Computer science students were recruited from student mailing lists of these universities; university lecturers advertised the experiment among their students. Students were rewarded with €50 vouchers for taking part in the experiment. The goal was to involve computer scientists and software engineers. Subjects were required to have completed or be in the process of completing a course on software design using UML.

Table 2.1 Dependent Variables of the experiment. (Abbreviations: Co = completeness; Ac = accuracy; S = state space; I = invariants; O = operations; PM = perceived modelling; PC = problem comprehension; PPC = perceived PC; DD = defect detection; MC = model comprehension; PDD = perceived DD; PMC = perceived MC; U = usefulness; EoU = ease of use; Us = usabiliy; R = reading; N = navigation; MOs = maps and overviews; LEC = live error checking; LF = look and feel; C = cohesion; L = learnability; CS = comfort/satisfaction; PN = preferred Notation; FN = future notation; NP = No preference)

Category	RQ	Kind	Description
Completeness (Co)	RQ1	Proportion	Measures satisfaction of requirements in produced model; this is a proportion (obtained score divided by maximum score). Satisfaction is measured on an ordinal scale from 4 (total satisfaction) to 1 (no satisfaction). **Variables**: CoS, CoI, CoO
Accuracy (Ac)	RQ1	Proportion	Model is tested against test cases and the satisfaction of each test case is scored. Satisfaction is measured on an ordinal scale: 4 (total satisfaction) to 1 (no satisfaction). **Variables**: AcS, AcI, AcO
Perceived Modelling (PM) Construction	RQ1	Nominal	Notation perceived as providing better modelling erformance; either: {VCL, UML, NP}. **Variables**: PMS, PMI, PMO
Problem Comprehension (PC)	RQ2	Proportion	Proportion of correct answers in the problem comprehension questionnaire. **Variables**: PC
Perceived Problem Comprehension (PPC)	RQ2	Nominal	Notation perceived as providing better PC performance; either: {VCL, UML, NP}. **Variables**: PPC
Model Comprehension and usage	RQ3	Proportion	Proportion of correct answers in model comprehension (MC) questionnaire or proportion of defects identified in model with seeded defects (DD). **Variables**: DD, MC
Perceived Model Comprehension and Usage	RQ3	Nominal	Notation perceived as providing better DD or MC performances; either: {VCL, UML, NP}. **Variables**: PDD, PMC
Usefulness and Ease of Use	RQ4	Proportion	Proportion calculated from score obtained by evaluating the responses to questions on a Likert Scale from 1 (strongly agree) to 5 (strongly disagree). **Variables**: U, EoU
Usability (Us)	RQ5	Nominal	Notation perceived as better for different usability criteria; either: {VCL, UML+OCL, NP}. **Variables**: UsR, UsN, UsMOs UsLEC, UsLF, UsC, UsL, UsCS
Appraisal (Appr)	RQ6	Nominal	Subject's appraisal of VCL. Comments of open questions are evaluated as {positive, negative, neutral} with respect to VCL. **Variables**: Appr
Preferred Notation (PN)	RQ6	Nominal	Preferred notation at state space (S), invariants (I), operations (O), overall and notation to use in the future; either: {VCL, UML, NP}. **Variables**: PNS, PNI, PNO, PN, FN

Table 2.2 Experiment's Hypotheses

Category	Null Hypothesis	Alternative Hypothesis
Completeness	H_1^0: $CoS(VCL) = CoS(UML)$	H_1^a: $CoS(VCL) > CoS(UML)$
	H_2^0: $CoI(VCL) = CoI(UML)$	H_2^a: $CoI(VCL) > CoI(UML)$
	H_3^0: $CoO(VCL) = CoO(UML)$	H_3^a: $CoO(VCL) > CoO(UML)$
Accuracy	H_4^0: $AcS(VCL) = AcS(UML)$	H_4^a: $AcS(VCL, S) > AcS(UML)$
	H_5^0: $AcI(VCL) = AcI(UML)$	H_5^a: $AcI(VCL) > AcI(UML)$
	H_6^0: $AcO(VCL) = AcO(UML)$	H_6^a: $AcO(VCL) > AcO(UML)$
Perceived	H_7^0: $PMS(VCL) = PMS(UML)$	H_7^a: $PMS(VCL) > PMS(UML)$
Modelling	H_8^0: $PMI(VCL) = PMS(UML)$	H_8^a: $PMS(VCL) > PMS(UML)$
Construction	H_9^0: $PMO(VCL) = PMO(UML)$	H_9^a: $PMO(VCL) > PMO(UML)$
Comprehension	H_{10}^0: $PC(VCL) = PC(UML)$	H_{10}^a: $PC(VCL) > PC(UML)$
and Model	H_{11}^0: $DD(VCL) = DD(UML)$	H_{11}^a: $DD(VCL) > DD(UML)$
Usage	H_{12}^0: $MC(VCL) = MC(UML)$	H_{12}^a: $MC(VCL) > MC(UML)$
Perceived	H_{13}^0: $PPC(VCL) = PPC(UML)$	H_{13}^a: $PPC(VCL) > PPC(UML)$
Comprehension	H_{14}^0: $PDD(VCL) = PDD(UML)$	H_{14}^a: $PDD(VCL) > PDD(UML)$
and Usage	H_{15}^0: $PMC(VCL) = PMC(UML)$	H_{15}^a: $PMC(VCL) > PMC(UML)$
Usefulnes and	H_{16}^0: $U(VCL) = U(UML)$	H_{16}^a: $U(VCL) > U(UML)$
Ease of Use	H_{17}^0: $EoU(VCL) = EoU(UML)$	H_{17}^a: $EoU(VCL) > EoU(UML)$
	H_{18}^0: $DU(VCL) = DU(UML)$	H_{18}^a: $DU(VCL) > DU(UML)$
	H_{19}^0: $DEoU(VCL) = DEoU(UML)$	H_{19}^a: $DEoU(VCL) > DEoU(UML)$
Usability	H_{20}^0: $UsR(VCL) = UsR(UML)$	H_{20}^a: $UsR(VCL) > UsR(UML)$
	H_{21}^0: $UsN(VCL) = UsN(UML)$	H_{21}^a: $UsN(VCL) > UsN(UML)$
	H_{22}^0: $UsMOs(VCL) = UsMOs(UML)$	H_{22}^a: $UsMOs(VCL) > UsMOs(UML)$
	H_{23}^0: $UsLEC(VCL) = UsLEC(UML)$	H_{23}^a: $UsLEC(VCL) > UsLEC(UML)$
	H_{24}^0: $UsLF(VCL) = UsLF(UML)$	H_{24}^a: $UsLF(VCL) > UsLF(UML)$
	H_{25}^0: $UsC(VCL) = UsC(UML)$	H_{25}^a: $UsC(VCL) > UsC(UML)$
	H_{26}^0: $UsL(VCL) = UsL(UML)$	H_{26}^a: $UsL(VCL) > UsL(UML)$
	H_{27}^0: $UsCS(VCL) = UsCS(UML)$	H_{27}^a: $UsCS(VCL) > UsCS(UML)$
Preferred	H_{28}^0: $PNS(VCL) = PNS(UML)$	H_{28}^a: $PNS(VCL) > PNS(UML)$
Notation	H_{29}^0: $PNI(VCL) = PNI(UML)$	H_{29}^a: $PNI(VCL) > PNI(UML)$
	H_{30}^0: $PNO(VCL) = PNO(UML)$	H_{30}^a: $PNO(VCL) > PNO(UML)$
	H_{31}^0: $PN(VCL) = PN(UML)$	H_{31}^a: $PN(VCL) > PN(UML)$
	H_{32}^0: $FN(VCL) = FN(UML)$	H_{32}^a: $FN(VCL) > FN(UML)$
Appraisal	H_{33}^0: $Appr(VCL) = Appr(UML)$	H_{33}^a: $Appr(VCL) > Appr(UML)$

Chapter 3

Experimental Setup

This chapter presents the design choices that underlie the experiment presented here.

3.1 Recruitment

The experiment was executed in four different venues: (a) Faculty of Sciences of the University of Lisbon, Portugal (FCUL); (b) Faculty of Science and Technology of the New University of Lisbon, Portugal (FCT/UNL); (c) University of Luxembourg (UL); and (d) University of York, UK (UY). Software engineering and computer science students were recruited from student mailing lists of these universities; university lecturers advertised the experiment in relevant classes. Students were rewarded with €50 vouchers for taking part in the experiment.

The goal was to involve computer scientists and software engineers. Subjects were required to have completed or be in the process of completing a course on object-oriented modelling using UML. The experiment's training amounted to 4 to 6 hours: two hours for each notation was mandatory and one extra hour for each notation was optional; it intended to complement the existing education of the participants to provide them with the means to carry out the tasks of the experiment competently. The training had a practical nature to prepare them for the challenging tasks of model construction in the experiment; it consisted of modelling a given system from a requirements description using the notations under study, VCL and UML+OCL, and the supporting tools, Visual Contract Builder (VCB) [AG15, AGK11][1] and Papyrus/UML[2].

3.2 Paired Comparison Design

This experiment involves one factor whose effect is of interest to us: the design modelling notation. This factor has two treatments: VCL and UML+OCL. The experiment is to use a *paired comparison* (*crossover*, or *within-subjects*) design [WRH+12]. This means that subjects will have to work on both systems. The rationale for doing this is as follows: (a) maximise the number of data points (observations) to increase statistical power and (b) ensure that the differences in complexity for the systems used in the tasks do not bias the results.

Our design is based on a randomised block design to account for the subjects' ability during data analysis. Subjects were therefore grouped into two blocks: *High Ability* and *Medium Ability*. This was done based on a questionnaire that assesses the ability of subjects in several aspects

[1] `http://vcl.gforge.uni.lu/`
[2] `http://www.eclipse.org/papyrus/`

that are important to the experiment, namely: (a) UML or object-oriented modelling ability, (b) OCL ability, (c) VCL ability, (d) ability in discrete mathematics and formal modelling. Based on these factors subjects were assessed on an ordinal scale from 1 to 5. Subjects with a score above 3.5 are placed in the *High Ability* block.

3.3 Tasks

The tasks of the experiment are designed to feed our dependent variables (table 2.1) to enable the statistical testing of our hypothesis. In total, subjects had to complete all the tasks in a session of two hours. The tasks are follows:

- *Model construction, state space and invariants.* This task feeds the dependent variables associated with completeness and accuracy of state space and invariants, namely, CoS, CoI, AcS and AcI. Subjects were given a requirements documents of a system to be modelled and a partial starting structural model that they were asked to complete; they were given 30 minutes to execute this task.

- *Model construction, operations.* This task, which followed the task on the model construction of state space and invariants, is associated with the dependent variables of completeness and accuracy of operations, namely, CoO and AcO. Subjects are given a complete structural model and they are are asked to fully specify two operations of the system. They were given 35 minutes to execute this task.

- *Problem comprehension.* Feeding the problem comprehension dependent variable (PC), this task is executed right after the model construction tasks to measure how well the subjects comprehended the problem and if there was any effect from the notations under study on comprehension. Subjects were asked to answer a questionnaire on problem comprehension within 15 minutes.

- *Defect Detection.* In this task, associated with the defect detection (DD) dependent variable, subjects were given a model of the case study with seeded defects and they were asked to identify as much defects as they could. The models contained 38 seeded defects and subjects were given 25 minutes to complete the task.

- *Model Comprehension.* In this task, associated with the model comprehension (MC) dependent variable, subjects were given a correct model of the case study and they were asked questions about it. Subjects were given 15 minutes to complete the task.

3.4 Instrumentation

Instrumentation deals with the means to perform and monitor a controlled experiment in order to derive quantitative results to be subject to a rigorous analysis.

The materials used in the experiment presented here comprise:

- requirements documents and sample models,

- questionnaires to assess problem and model comprehension,

- de-briefing survey.

These are detailed in the sections that follow.

3.4.1 Requirements documents and sample models

The tasks of the experiment were designed around the following two case studies, which are described in detail in appendix B:

- *University Library.* This is a system of a typical university Library. It enables users to borrow copies of books. There are three kinds of users: *students*, *research students* and *academic staff*. Each kind of user has different borrowing entitlements.
- *Flight Booking.* This a simple system to manage flight reservations.

Subjects were given out the descriptions of the case studies in appendix B to model at the start of each session.

Chapter 4

Experimental Results

VCL's goal is to facilitate software design modelling through the use of visualisation. The controlled experiment presented here investigates the effectiveness of VCL as a visual notation by comparing it with the standard modelling notations UML and OCL. This chapter presents and discusses the results of the controlled experiment.

4.1 Preliminaries

4.1.1 On the Statistical Analysis

The experiment is designed to enable Null Hypothesis Significance Testing (NHST). Following guidelines advocated by methodologists, influential journals and leading figures in social science research, we supplement our NHST-based analysis with effect sizes (ESs) and confidence intervals. ESs are quantitative estimates of the magnitude of some effect of interest; often, they quantify the size of a difference between two groups. Confidence intervals (CIs) indicate the precision of a point estimate or measurement, providing a range of plausible values within which we can have a degree of confidence that the estimate is not due to chance.

The statistical analysis presented in the sequel has been carried out using the R statistical software [R C15]. R was used to perform statistical calculations and generate graphs describing data.

Means, Proportions and their CIs

Our analysis emphasises measures of central tendency. For continuous variables, we rely on the mean (or average) taken from samples of a population and constituting point estimates. To enhance the precision of such estimates, we calculate the 95% CI interval of a mean, in terms of its lower and upper bounds; this is calculated using the classical formula [Cum12]:

$$95\%CI = M \pm 1.96 \times SE \quad SE = \frac{SD}{\sqrt{N}}$$

Above: M is the sample mean, SE stands for standard error, SD is the standard deviation, and N is the size of the sample. The value of 1.96 corresponds to the Z score representing the 95% range in the normal distribution.

Categorical variables are scrutinised based on proportions, a measurement akin to the mean. A proportion is the fraction of a number of discrete things that have a property of interest [Cum12] – if x is the number of such things, and N is the total number of things, the proportion is

$P = x/N$. We derive proportions from frequency distributions. Like means, proportions are point estimates that can be made precise by providing a CI. To calculate CIs for proportions, we use the more robust methods of Newcombe et al [New98, NA00], as recommended by [Cum12], which provide good approximations even when N is small and P is close or equal to 0 or 1. This is described by the following formula[1]:

$$A = 2 \times x + 1.96^2 \quad B = 1.96 \times \sqrt{1.96^2 + 4 \times x \times (1 - P)} \quad C = 2 \times (N + 1.96^2)$$
$$95\%CI = [(A - B)/C, (A + B)/C]$$

Where P is the estimated proportion, x is the observed frequency for the property of interest, and N is the total number of things – $P = x/N$.

Hypothesis Testing

The experiment's dependent variables are given in table 2.1 (chapter 2); the corresponding hypothesis are formulated in table 2.2. For a dependent variable V, the null and alternative hypothesis are formulated as:

$$H_i^0 : V(VCL) = V(UML + OCL) \quad H_i^a : V(VCL) > V(UML + OCL)$$

Hypothesis are tested by estimating probabilities, known as p-values. Given data D, and some null hypothesis H_0, a p-value is an estimate for the conditional probability $P(D \mid H_0)$ [Coh94] – the likelihood of the given observations assuming that the null hypothesis is true —; hence, when we reject H_0, we are saying that it is unlikely because our observations deems $P(D \mid H_0)$ as unlikely – in this case, we accept the alternative hypothesis. We consider three levels of statistical significance to enable the rejection of the null hypothesis and the acceptance of a statistically significant difference; depending on whether the p-value is below $\alpha = 0.05$ (*), $\alpha = 0.01$ (**) or $\alpha = 0.001$ (***). For continuous variables, we use the non-parametric Wilcoxon tests (we give the p-values estimated from the t-test in the text also); categorical variables are hypothesis tested with Z tests.

Effect Sizes

An ES measurement used in our analysis is the raw (or unstandardised) mean difference. Given the experiment's within-subjects design, we need to consider that the variables are dependent and are often the result of repeated measures and distinguish this more complicated case with the simpler between-subjects design, where the variables are independent, and the calculation is, as a result, rather more straightforward [GK05, PJ13, FL12]. We work with paired differences; the formulas are as follows [GK05, PJ13, FL12, Cum12][2]:

$$PD = V_{VCL} - V_{UML} \quad SE_{PD} = \frac{SD_{PD}}{\sqrt{N}} \quad 95\%CI = M_{PD} \pm 1.96 \times SE_{PD}$$

Above, we have that $M_{PD} = M_{VCL} - M_{UML}$; SD_{PD} is standard deviation of the paired differences.

Raw mean differences provide an intuitive measurement of an effect, but they lack uniformity, making it difficult to compare effects when the scales differ. To provide a more uniform measurement, we use the ES Cohen's d [Coh88], which is appropriate for continuous variables

[1] The traditional formula to calculate SEs for CIs of such proportions is $SE = \sqrt{\frac{P \times (1-P)}{N}}$

[2] We make a slight adjustment to the formulas used by these authors; we use the coefficient with value 1.96 derived from the normal distribution; these authors use the t distribution and the value $t_{N-1}(.975)$, corresponding to the 95% range.

with an underlying analysis centred on means. To estimate this, we use the standard deviation average (or pooled standard deviation) as the standardiser (denominator) [Cum12]:

$$d = \frac{M_{VCL} - M_{UML}}{SD_{av}} \quad SD_{av} = \sqrt{\frac{SD^2_{VCL} + SD^2_{UML}}{2}}$$

There are slightly different ways of calculating the Cohen's d, which vary depending on the formula used for the denominator. The formula above, based on the standard deviation average, fits the paired design being followed [Cum12]. CIs for this ES are calculated using approaches based on non-central t distributions [CF01, Kel07]; in particular we use the R package MBESS [Kel07] for this effect.

For categorical data, we use the ES measurement Cohen's h [Coh88], based on the arcsine transformation, which is appropriate for differences between proportions. The formula for h and the SE to calculate CIs (from [Coh88]) is[3]:

$$h = 2 \times \arcsin\sqrt{P_{VCL}} - 2 \times \arcsin\sqrt{P_{UML}} \quad SE = \frac{\sqrt{1}}{N}$$

4.1.2 On Graphical representations

The results of this chapter are represented using various kinds of graphs. These are as follows:

- *Box plots* (example in Fig. 4.1a) present the upper and lower quartiles of the data by the top and bottom of the rectangle, and the median by the horizontal line within the rectangle; the lines extending vertically from the boxes (whiskers) indicate variability outside the upper and lower quartiles. The symbol `o` represents outliers and * indicates the mean. These plots display the different samples in the abscissa (x-axis) and the units of the analysed data in the ordinate (y-axis) — Fig. 4.1a uses score rates in the ordinate.
- *Plot of point estimates and confidence intervals* (example in Fig. 4.1b) portrays point estimates (e.g. means) as dots and 95% confidence intervals (CIs) as error bars. They display the different samples in the abscissa and the units of the analysed data in the ordinate (Fig. 4.1b uses score rates).
- *Forest plots* (example in Fig. 4.7a) display point estimates and their corresponding 95% CIs. The point estimate is represented as a circles and the CI as error bars; the abcissa contains the scale corresponding to the presented result.
- *Histograms* (example in Fig. 4.6), a variant of bar chart, uses the area of bars to represent the proportion of a phenomena accounted by the corresponding category. Histograms are typically used to depict frequency distributions.

4.2 Results: a Bird's Eye View

The outcomes of hypothesis testing are summarised in table 4.1. The rows indicate the confirmed hypothesis: either the null hypothesis (no difference in the approaches) or the alternative hypothesis (a significant difference in favour of VCL enabled the rejection of the null hypothesis). The confirmed alternative hypothesis are highlighted in green. The μ columns present the means of the scores underlying the corresponding hypothesis. The *p-value* column reports on the statistical significance of the differences between the examined approaches. The ES column indicates the effect size, a quantitative measure of the strength of the presented results.

[3]Cohen [Coh88] defines the variance for our experimental setting as $1/N$.

Table 4.1 Results of Hypothesis testing. (Cell colours: confirmed alternative hypothesis in shaded (green in colour version), and null hypothesis in white. Statistical significance: * significant with $\alpha = .05$, ** $\alpha = .01$, *** $\alpha = .001$, ns = not significant. Effect magnitude: ø is null, • is small, •• is medium, ••• is large, •+ is small to medium, ••+ is medium to large.)

Hypothesis	$VCL_{M/P}$	$UML_{M/P}$	VCL - UML	p-value	ES(d/h)
Model construction (objective)					
H_1^0 : *CoS*	$.64_M[.6, .69]$	$.67_M[.63, .71]$	$-.03[-.07, .01]$	$.89_{ns}$	$-.13^{\bullet}_{d}[-.35, .08]$
H_2^0 : *CoI*	$.1_M[.07, .13]$	$.13_M[.09, .18]$	$-.03[-.08, .01]$	$.84_{ns}$	$-.18^{\bullet}_{d}[-.38, .05]$
H_3^a : *CoO*	$.16_M[.13, .19]$	$.11_M[.09, .13]$	$.05[.03, .08]$	$2 \times 10^{-6}_{***}$	$.45^{\bullet\bullet}_{d}[.26, .71]$
H_4^0 : *AcS*	$.38_M[.33, .43]$	$.38_M[.34, .43]$	$-.001[-.05, .05]$	$.59_{ns}$	$-.01^{ø}_{d}[-.22, .21]$
H_5^0 : *AcI*	$.2_M[.16, .24]$	$.2_M[.15, .24]$	$.01[-.04, .05]$	$.37_{ns}$	$.03^{ø}_{d}[-.19, .24]$
H_6^a : *AcO*	$.11_M[.08, .14]$	$.09_M[.06, .11]$	$.02[-.003, .04]$	$.035_{*}$	$.17^{\bullet}_{d}[-.03, .4]$
Perceived model construction (subjective)					
H_7^0 : *PMS*	$.51_P[.37, .65]$	$.35_P[.22, .5]$	$.16[-.11, .41]$	$.064_{ns}$	$.33^{\bullet+}_{h}[.03, .63]$
H_8^a : *PMI*	$.53_P[.39, .67]$	$.26_P[.15, .4]$	$.28[.02, .5]$	$.0041_{**}$	$.58^{\bullet\bullet}_{h}[.28, .88]$
H_9^a : *PMO*	$.6_P[.46, .74]$	$.28_P[.17, .43]$	$.33[.05, .55]$	$.0012_{**}$	$.67^{\bullet\bullet+}_{h}[.37, .97]$
Comprehension and model usage (objective)					
H_{10}^0 : *PC*	$.65_M[.61, .69]$	$.64_M[.6, .68]$	$.01[-.04, .05]$	$.33_{ns}$	$.03^{ø}_{d}[-.18, .24]$
H_{11}^a : *DD*	$.27_M[.24, .3]$	$.19_M[.17, .21]$	$.08[.05, .1]$	$2 \times 10^{-7}_{***}$	$.67^{\bullet\bullet+}_{d}[.41, .87]$
H_{12}^a : *MC*	$.67_M[.64, .71]$	$.61_M[.58, .64]$	$.06[.02, .1]$	$.0012_{**}$	$.37^{\bullet+}_{d}[.09, .52]$
Perceived comprehension and model usage (subjective)					
H_{13}^a : *PPC*	$.47_P[.33, .61]$	$.12_P[.05, .24]$	$.35[.13, .53]$	$.00018_{***}$	$.81^{\bullet\bullet\bullet}_{h}[.51, 1.1]$
H_{14}^a : *PDD*	$.56_P[.41, .7]$	$.23_P[.13, .38]$	$.33[.06, .54]$	$.001_{**}$	$.68^{\bullet\bullet+}_{h}[.38, .98]$
H_{15}^a : *PMC*	$.44_P[.3, .59]$	$.14_P[.07, .27]$	$.3[.08, .49]$	$.001_{**}$	$.69^{\bullet\bullet+}_{h}[.39, .99]$
Usefulness and ease of use (subjective)					
H_{16}^0 : *U*	$.75_M[.71, .78]$	$.72_M[.69, .76]$	$.02[-.03, .07]$	$.095_{ns}$	$.18^{\bullet}_{d}[-.18, .42]$
H_{17}^a : *EoU*	$.66_M[.63, .69]$	$.6_M[.56, .63]$	$.07[.03, .11]$	$.0026_{**}$	$.64^{\bullet\bullet+}_{d}[.16, .79]$
H_{18}^0 : *DU*	$.28_P[.17, .43]$	$.21_P[.11, .35]$	$.07[-.14, .27]$	$.23_{ns}$	$.16^{\bullet}_{h}[-.14, .46]$
H_{19}^a : *DEoU*	$.58_P[.43, .72]$	$.23_P[.13, .38]$	$.35[.08, .56]$	$.0005_{***}$	$.73^{\bullet\bullet+}_{h}[.43, 1.03]$
Usability (subjective)					
H_{20}^a : *UsR*	$.53_P[.39, .67]$	$.33_P[.2, .47]$	$.21[-.07, .45]$	$.025_{*}$	$.43^{\bullet\bullet}_{h}[.13, .73]$
H_{21}^a : *UsN*	$.69_P[.54, .81]$	$.14_P[.07, .28]$	$.55[.29, .72]$	$2 \times 10^{-7}_{***}$	$1.19^{\bullet\bullet\bullet}_{h}[.88, 1.49]$
H_{22}^a : *UsMOs*	$.69_P[.54, .81]$	$.14_P[.07, .28]$	$.55[.29, .72]$	$2 \times 10^{-7}_{***}$	$1.19^{\bullet\bullet\bullet}_{h}[.88, 1.49]$
H_{23}^a : *UsLEC*	$.51_P[.37, .65]$	$.09_P[.04, .22]$	$.42[.2, .59]$	10^{-5}_{***}	$.97^{\bullet\bullet\bullet}_{h}[.68, 1.27]$
H_{24}^a : *UsLF*	$.7_P[.55, .81]$	$.16_P[.08, .3]$	$.53[.27, .72]$	$3 \times 10^{-7}_{***}$	$1.15^{\bullet\bullet\bullet}_{h}[.85, 1.45]$
H_{25}^0 : *UsC*	$.37_P[.24, .52]$	$.27_P[.16, .42]$	$.1[-.14, .32]$	$.17_{ns}$	$.21^{\bullet+}_{h}[-.1, .52]$
H_{26}^0 : *UsL*	$.46_P[.3, .64]$	$.32_P[.18, .51]$	$.14[-.18, .43]$	$.14_{ns}$	$.29^{\bullet+}_{h}[-.08, .66]$
H_{27}^a : *UsCS*	$.54_P[.36, .7]$	$.25_P[.13, .43]$	$.29[-.04, .55]$	$.014_{*}$	$.6^{\bullet\bullet}_{h}[.22, .97]$
Preferred notation (subjective)					
H_{28}^0 : *PNS*	$.42_P[.28, .57]$	$.44_P[.3, .59]$	$-.02[-.29, .24]$	$.59_{ns}$	$-.05^{ø}_{h}[-.35, .25]$
H_{29}^a : *PNI*	$.65_P[.5, .78]$	$.19_P[.1, .33]$	$.47[.2, .66]$	$6 \times 10^{-6}_{***}$	$.99^{\bullet\bullet\bullet}_{h}[.69, 1.28]$
H_{30}^a : *PNO*	$.58_P[.43, .72]$	$.21_P[.11, .35]$	$.37[.11, .58]$	$.00021_{***}$	$.78^{\bullet\bullet+}_{h}[.49, 1.08]$
H_{31}^a : *PN*	$.49_P[.35, .63]$	$.28_P[.17, .43]$	$.21[-.05, .44]$	$.023_{*}$	$.43^{\bullet\bullet}_{h}[.14, .73]$
H_{32}^0 : *FN*	$.33_P[.2, .47]$	$.33_P[.2, .47]$	$0[-.23, .23]$	$.5_{ns}$	$0^{ø}_{h}[-.3, .3]$
H_{33}^a : *Appr*	$.56_P[.51, .61]$	$.31_P[.27, .36]$	$.24[.15, .33]$	$4 \times 10^{-12}_{***}$	$.5^{\bullet\bullet}_{h}[.4, .6]$

The controlled experiment presented here adopted a crossover design. This means that subjects had two attempts at executing the different tasks based on two different case studies: the university library case study (UL) and the flight booking case study (FB). Table 4.1 documents the results at each case study and the collective results of both case studies. The table results that do not give values for the individual case studies are based on the debriefing questionnaire.

The presentation of the results is organised around: *model construction*, *comprehension and model usage*, *usefulness and ease of use*, *usability* and *preferred notations*.

4.3 Model Construction

The performance of subjects in performing model construction tasks is evaluated objectively based on two different criteria:

- *Completeness*, which measures how much has been modelled;
- and *accuracy*, which assesses the quality and meaning of the produced models.

We conduct a subject investigation on how subjects perceived their model construction performances. We present the data for each evaluation criterium and model facet using the following graphical representations: one box plot and two plots of point estimates and CIs, one focussed on means and the other on mean differences (e.g. Fig. 4.1). These graphs present the results in total, for each case study – university library or flight booking –, and for each group or institution.

4.3.1 Completeness

The completeness results of the three modelling facets – state space, invariants and operations –, corresponding to hypothesis $H_{1..3}$ of table 4.1, are summarised in table 4.2 and portrayed in Fig. 4.1.

State Space

Figures 4.1a, 4.1b and 4.1c portray the results of completeness of state space, summarised in table 4.4a. We can see that there are variations of performance across the different samples, but the differences remain nearly the same, highlighting a quasi-equality that confirms the null hypothesis: $CoS(VCL) = CoS(UML + OCL)$. The mean of VCL's performance rate is .64 (95% CI $[.6, .69]$, $sd = .22$) and that of UML+OCL is slightly higher .67 (CI $[.63, .71]$, sd=.19), with a mean of differences of $-.03$ (CI $[-.07, .01]$). Cohen's d ES measurement of $-.13$ (CI $[-.35, .08]$, •) indicates a small effect.

Invariants

The results portrayed in Figs. 4.1d, 4.1e and 4.1f highlight essentially similar performances, confirming the null hypothesis: $CoI(VCL) = CoI(UML + OCL)$. It is interesting to observe that in the sample FCT/UNL UML+OCL obtains a higher result, which is not confirmed by the remaining samples. The mean of VCLs performance rate was .1 (CI$[.07, .13]$, sd = .14) and that of UML+OCL was slightly higher .13 (CI$[.09, .18]$, sd = .22) with a mean of differences of $-.03$ (CI$[-.08, .01]$). Cohen's d ES measurement of $-.18$ (CI$[-.38, .05]$, •) denotes a small effect.

	University Library							Flight Booking							Total						
	N	VCL		UML		PD		N	VCL		UML		PD		N	VCL		UML		PD	
		M	S	M	S	M	S		M	S	M	S	M	S		M	S	M	S	M	S
(1)	16	.53[.43, .63]	.2	.56[.46, .65]	.19	−.03[−.14, .08]	.23	16	.54[.42, .66]	.24	.58[.47, .68]	.22	−.04[−.12, .05]	.18	32	.53[.46, .61]	.22	.57[.5, .64]	.2	−.03[−.1, .04]	.2
(2)	10	.75[.61, .88]	.22	.8[.72, .88]	.13	−.05[−.15, .05]	.16	11	.85[.81, .9]	.08	.79[.71, .87]	.13	.06[−.01, .13]	.12	21	.8[.73, .87]	.17	.79[.74, .85]	.13	.01[−.05, .07]	.15
(3)	9	.57[.46, .68]	.17	.67[.6, .74]	.11	−.1[−.25, .05]	.23	9	.66[.5, .81]	.24	.64[.5, .79]	.22	.01[−.16, .18]	.26	18	.61[.52, .71]	.21	.66[.58, .74]	.17	−.04[−.16, .07]	.25
(3)	7	.69[.6, .78]	.12	.72[.63, .81]	.13	−.03[−.13, .06]	.13	7	.72[.56, .87]	.21	.76[.7, .83]	.09	−.05[−.19, .09]	.19	14	.7[.62, .79]	.16	.74[.69, .8]	.11	−.04[−.12, .04]	.16
(4)	42	.62[.55, .68]	.2	.67[.61, .72]	.18	−.05[−.11, .01]	.2	43	.67[.6, .74]	.23	.68[.62, .74]	.2	−.003[−.06, .05]	.19	85	.64[.6, .69]	.22	.67[.63, .71]	.19	−.03[−.07, .01]	.19

(a) State Space (variable *CoS*)

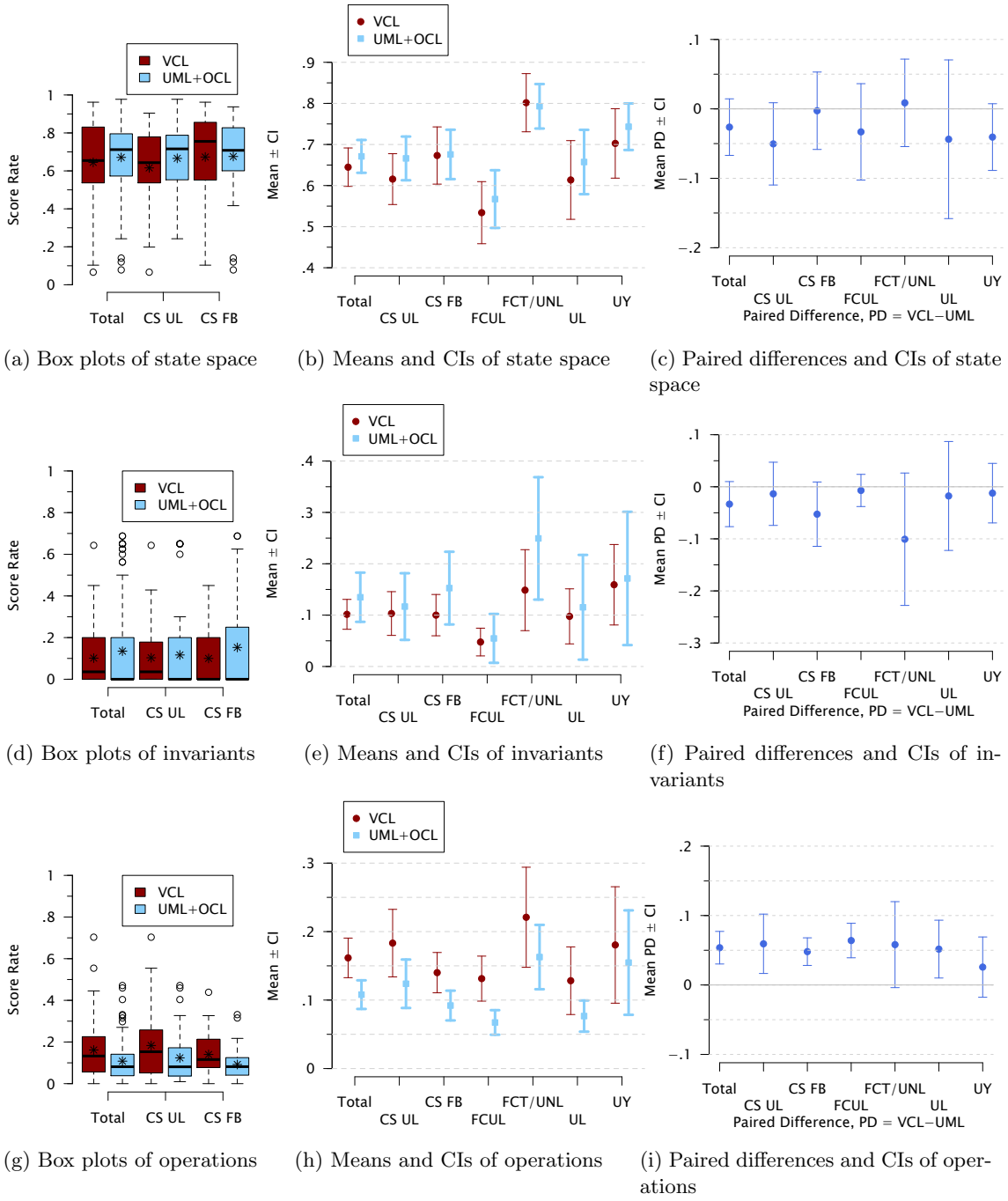
	University Library							Flight Booking							Total						
	N	VCL		UML		PD		N	VCL		UML		PD		N	VCL		UML		PD	
		M	S	M	S	M	S		M	S	M	S	M	S		M	S	M	S	M	S
(1)	16	.05[.01, .08]	.08	.03[−.01, .07]	.08	.01[−.01, .04]	.05	16	.05[.01, .09]	.08	.08[−.01, .17]	.18	−.03[−.08, .03]	.11	32	.05[.02, .07]	.08	.05[.01, .1]	.14	−.01[−.04, .02]	.09
(2)	10	.16[.02, .29]	.21	.23[.05, .41]	.29	−.07[−.27, .13]	.32	11	.14[.04, .24]	.16	.27[.1, .43]	.28	−.13[−.3, .04]	.29	21	.15[.07, .23]	.18	.25[.13, .37]	.28	−.1[−.23, .03]	.3
(3)	9	.1[.04, .15]	.08	.11[−.04, .25]	.23	−.01[−.15, .13]	.22	9	.1[.003, .2]	.15	.12[−.02, .27]	.23	−.02[−.19, .14]	.25	18	.1[.04, .15]	.12	.12[.01, .22]	.22	−.02[−.12, .09]	.23
(4)	7	.17[.05, .28]	.16	.16[−.02, .35]	.25	.004[−.15, .16]	.21	7	.15[.03, .27]	.16	.18[−.02, .38]	.27	−.03[−.16, .1]	.18	14	.16[.08, .24]	.15	.17[.04, .3]	.25	−.01[−.11, .09]	.19
(5)	42	.1[.06, .15]	.14	.12[.05, .18]	.21	−.01[−.07, .05]	.2	43	.1[.06, .14]	.13	.15[.08, .22]	.24	−.05[−.11, .01]	.21	85	.1[.07, .13]	.14	.13[.09, .18]	.23	−.03[−.08, .01]	.2

(b) Invariants (variable *CoI*)

	University Library							Flight Booking							Total						
	N	VCL		UML		PD		N	VCL		UML		PD		N	VCL		UML		PD	
		M	S	M	S	M	S		M	S	M	S	M	S		M	S	M	S	M	S
(1)	16	.15[.1, .2]	.1	.07[.04, .09]	.05	.09[.04, .13]	.09	16	.11[.07, .15]	.08	.07[.04, .1]	.06	.04[.02, .06]	.04	32	.13[.1, .16]	.09	.07[.05, .09]	.05	.06[.04, .09]	.07
(2)	11	.26[.14, .39]	.21	.17[.1, .25]	.13	.09[−.02, .2]	.19	11	.18[.11, .25]	.12	.15[.09, .21]	.1	.03[−.02, .08]	.08	22	.22[.15, .29]	.17	.16[.12, .21]	.11	.06[0, .12]	.15
(3)	9	.13[.05, .21]	.13	.09[.05, .13]	.06	.04[−.03, .11]	.11	9	.13[.07, .19]	.09	.06[.04, .09]	.03	.06[.02, .11]	.07	18	.13[.08, .18]	.11	.08[.05, .1]	.05	.05[.01, .09]	.09
(4)	7	.2[.04, .36]	.21	.22[.09, .36]	.18	−.03[−.16, .11]	.18	7	.16[.09, .24]	.1	.09[.05, .12]	.05	.08[.02, .13]	.08	14	.18[.1, .27]	.16	.15[.08, .23]	.15	.03[−.05, .1]	.15
(5)	43	.18[.13, .23]	.16	.12[.09, .16]	.12	.06[.02, .1]	.14	43	.14[.11, .17]	.1	.09[.07, .11]	.07	.05[.03, .07]	.07	86	.16[.13, .19]	.14	.11[.09, .13]	.1	.05[.03, .08]	.11

(c) Operations (variable *CoO*)

Table 4.2: Experimental data on completeness. ((1) = FCUL; (2) = FCT/UNL; (3) = U. Luxembourg; (4) = U. York; (5) = Total)



(a) Box plots of state space (b) Means and CIs of state space (c) Paired differences and CIs of state space

(d) Box plots of invariants (e) Means and CIs of invariants (f) Paired differences and CIs of invariants

(g) Box plots of operations (h) Means and CIs of operations (i) Paired differences and CIs of operations

Figure 4.1: Model completeness for state space, invariants and operations. (CS UL = University Library case study (CS); CS FB = Flight Booking case study; FCUL, FCT/UNL, UL and UY are the different groups/institutions)

Operations

Figures 4.1g, 4.1h and 4.1i highlight a clear advantage in favour of VCL, which, despite variations, is consistent across the four samples. The mean of VCL's performance rate is .16 ($sd = .14$, CI $[.13, .19]$) and that of $UML + OCL$ is .11 ($sd = .1$, CI $[.09, .13]$) with a mean difference of .05 (CI $[.03, .08]$). This difference is significant – Wilcoxon test p-value is $10^{-6} < 0.001$ (***) (t-test p-value = 10^{-5}) – to confirm the alternative hypothesis: $CoO(VCL) > CoO(UML + OCL)$. The ES measurement Cohen's d of .45 (CI $[.26, .71]$, ••) indicates a medium effect.

4.3.2 Accuracy

The accuracy results, corresponding to the hypothesis $H_{4..6}$ of table 4.1, are summarised in table 4.3 and portrayed in Fig. 4.2; they are discussed in detail in the sequel.

State Space

Figures 4.2a, 4.2b and 4.2c highlights performances that are quasi-equal, which confirms the null-hypothesis: $AcS(VCL) = AcS(UML + OCL)$. VCL's mean performance rate lies at .38 ($sd = .23$, CI $[.33, .43]$) and that of UML at .38 ($sd = .22$, CI $[.34, .43]$) with a mean difference of 0 (CI $[-.05, .05]$). The effect size measurement Cohen's d of $-.01$ (CI $[-.22, .21]$, ø) signals a null effect and endorses the quasi-equality of the results.

Invariants

The same quasi-equality effect, portrayed in Figs. 4.2d, 4.2e and 4.2f, is confirmed by the null-hypothesis: $AcI(VCL) = AcI(UML + OCL)$. VCL has a mean performance rate of .2 ($sd = .18$, CI $[.16, .24]$), the same as $UML + OCL$'s ($sd = .21$, CI $[.15, .24]$) with a minor difference of .01 (CI $[-.04, .05]$). Cohen's d ES measurement of .03 (CI $[-.19, .24]$, ø) stipulates a null effect.

Operations

Figures 4.2g, 4.2h and 4.2f highlight a difference in favour of VCL, which is statistically significant: $AcO(VCL) > AcO(UML + OCL)$. VCL's mean rate is .11 ($sd = .13$, CI $[.08, .14]$) and that of UML is .09 ($sd = .11$, CI $[.06, .11]$) with a mean difference of .02 (CI $[0, .04]$). The p-value of .035, obtained from Wilcoxon test (t-test p-value = .047), signals a statistically significant difference at the lowest level of significance ($p < .05$, *). Cohen's d ES measurement of .17 (CI $[-.03, .4]$, ••) signals a small effect.

4.3.3 Perceived Performance

The debriefing questionnaire (appendix E, section E.5) asked subjects whether they felt that any of the languages under scrutiny, VCL or UML+OCL, gave them any advantage in the model constructions tasks. The actual questions can be found in the section *The modelling tasks* in the questionnaire of section E.5. The results of this subjective assessment, corresponding to hypothesis $H_{7..9}$, are given Fig. 4.3, which contains a table of frequencies (Fig. 4.3a), a bar chart (Fig. 4.3b) and plot of proportions and their CIs (Fig. 4.3c). The results are as follows:

- In state-space modelling, most subjects felt that they performed better using VCL (22 out 43= .51, CI $[.39, .67]$), followed by $UML + OCL$ (15/43 = .35, CI $[.15, .4]$); 6 subjects felt that they performed equally using both approaches (*no preference* option). The difference of proportions between VCL and UML+OCL is .16 (CI $[-.11, .41]$). This was deemed

	University Library							Flight Booking							Total						
	N	VCL		UML		PD		N	VCL		UML		PD		N	VCL		UML		PD	
		M	S	M	S	M	S		M	S	M	S	M	S		M	S	M	S	M	S
(1)	16	.34[.24, .44]	.21	.3[.2, .4]	.19	.04[−.06, .14]	.2	16	.2[.12, .28]	.16	.2[.13, .27]	.15	0[−.08, .08]	.17	32	.27[.2, .34]	.2	.25[.19, .31]	.18	.02[−.04, .08]	.18
(2)	10	.6[.47, .73]	.22	.56[.47, .66]	.16	.04[−.1, .17]	.21	11	.39[.26, .51]	.21	.49[.38, .61]	.19	−.11[−.27, .06]	.28	21	.49[.39, .59]	.24	.53[.45, .6]	.18	−.04[−.15, .07]	.25
(3)	9	.42[.26, .59]	.25	.37[.27, .47]	.15	.05[−.19, .3]	.38	9	.3[.22, .38]	.12	.31[.2, .42]	.16	−.01[−.14, .13]	.2	18	.36[.27, .46]	.2	.34[.27, .41]	.16	.02[−.11, .15]	.3
(4)	7	.69[.6, .78]	.12	.72[.63, .81]	.13	−.03[−.13, .06]	.13	7	.34[.22, .46]	.16	.36[.23, .49]	.17	−.02[−.13, .08]	.14	14	.51[.39, .63]	.23	.54[.42, .66]	.24	−.03[−.09, .04]	.13
(5)	42	.48[.4, .55]	.25	.45[.38, .52]	.23	.03[−.04, .1]	.24	43	.29[.24, .35]	.18	.32[.26, .38]	.2	−.03[−.09, .03]	.2	85	.38[.33, .43]	.23	.38[.34, .43]	.22	0[−.05, .05]	.22

(a) State Space (variable AcS)

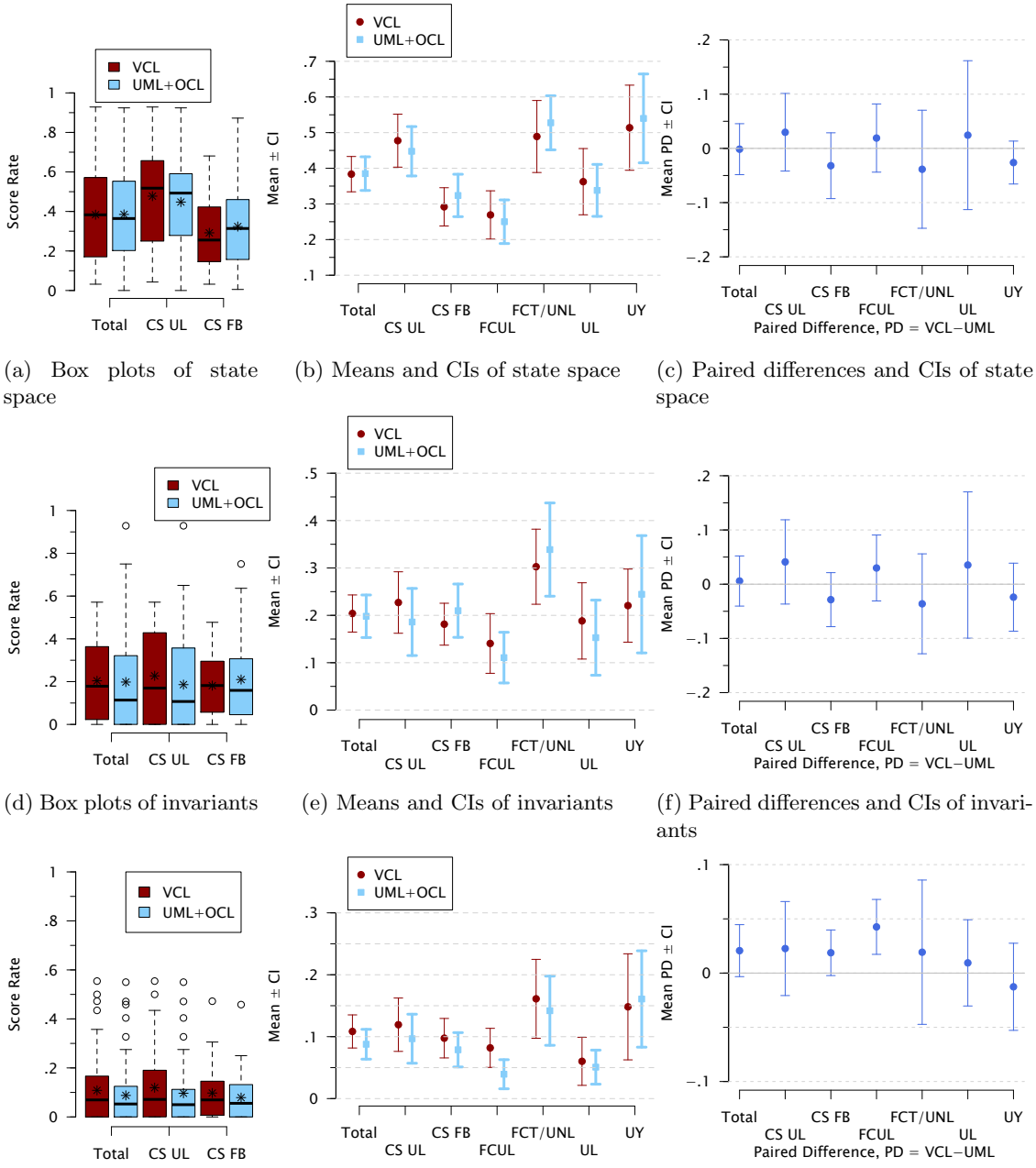
	University Library							Flight Booking							Total						
	N	VCL		UML		PD		N	VCL		UML		PD		N	VCL		UML		PD	
		M	S	M	S	M	S		M	S	M	S	M	S		M	S	M	S	M	S
(1)	16	.19[.09, .29]	.21	.13[.04, .22]	.18	.06[−.04, .15]	.19	16	.09[.02, .16]	.14	.09[.03, .16]	.13	0[−.07, .08]	.16	32	.14[.08, .2]	.18	.11[.06, .16]	.15	.03[−.03, .09]	.18
(2)	10	.34[.2, .49]	.23	.37[.2, .53]	.26	−.02[−.18, .14]	.26	11	.26[.19, .34]	.13	.31[.19, .43]	.2	−.05[−.15, .06]	.18	21	.3[.22, .38]	.19	.34[.24, .44]	.23	−.04[−.13, .06]	.22
(3)	9	.21[.06, .36]	.23	.1[−.04, .24]	.21	.11[−.14, .36]	.38	9	.16[.1, .23]	.1	.2[.13, .28]	.11	−.04[−.14, .06]	.15	18	.19[.11, .27]	.17	.15[.07, .23]	.17	.04[−.1, .17]	.29
(4)	7	.17[.05, .28]	.16	.16[−.02, .35]	.25	0[−.15, .16]	.21	7	.27[.18, .37]	.13	.32[.17, .48]	.21	−.05[−.21, .11]	.22	14	.22[.14, .3]	.15	.24[.12, .37]	.24	−.02[−.13, .08]	.21
(5)	42	.23[.16, .29]	.21	.19[.12, .26]	.23	.04[−.04, .12]	.26	43	.18[.14, .23]	.15	.21[.15, .27]	.19	−.03[−.08, .02]	.17	85	.2[.16, .24]	.18	.2[.15, .24]	.21	.01[−.04, .05]	.22

(b) Invariants (variable AcI)

	University Library							Flight Booking							Total						
	N	VCL		UML		PD		N	VCL		UML		PD		N	VCL		UML		PD	
		M	S	M	S	M	S		M	S	M	S	M	S		M	S	M	S	M	S
(1)	16	.09[.04, .14]	.09	.03[0, .07]	.06	.06[.02, .1]	.08	16	.07[.03, .12]	.09	.04[.01, .08]	.07	.03[0, .06]	.06	32	.08[.05, .11]	.09	.04[.02, .06]	.07	.04[.02, .07]	.07
(2)	11	.16[.06, .26]	.17	.15[.06, .24]	.15	.01[−.11, .14]	.21	11	.16[.07, .24]	.14	.13[.06, .21]	.12	.02[−.03, .08]	.09	22	.16[.1, .22]	.15	.14[.09, .2]	.13	.02[−.05, .09]	.16
(3)	9	.06[0, .11]	.09	.04[.01, .08]	.06	.01[−.06, .08]	.11	9	.06[.01, .12]	.08	.06[.01, .1]	.07	.01[−.03, .05]	.06	18	.06[.02, .1]	.08	.05[.02, .08]	.06	.01[−.03, .05]	.09
(4)	7	.2[.04, .36]	.21	.22[.09, .36]	.18	−.03[−.16, .11]	.18	7	.1[.04, .16]	.08	.1[.04, .16]	.08	0[−.05, .05]	.07	14	.15[.06, .23]	.16	.16[.08, .24]	.15	−.01[−.08, .06]	.13
(5)	43	.12[.08, .16]	.14	.1[.06, .14]	.13	.02[−.02, .07]	.15	43	.1[.07, .13]	.11	.08[.05, .11]	.09	.02[0, .04]	.07	86	.11[.08, .14]	.13	.09[.06, .11]	.11	.02[0, .04]	.11

(c) Operations (variable AcO)

Table 4.3: Experimental data on accuracy. ((1) = FCUL; (2) = FCT/UNL; (3) = U. Luxembourg; (4) = U. York; (5) = Total)



(a) Box plots of state space

(b) Means and CIs of state space

(c) Paired differences and CIs of state space

(d) Box plots of invariants

(e) Means and CIs of invariants

(f) Paired differences and CIs of invariants

(g) Box plots of operations

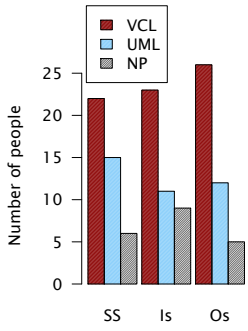
(h) Means and CIs of operations

(i) Paired differences and CIs of operations

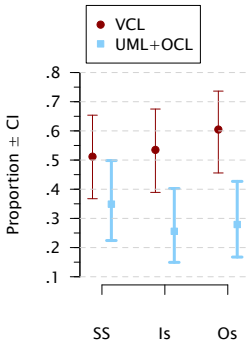
Figure 4.2: Model accuracy for state space, invariants and operations. (CS UL = University Library case study (CS); CS FB = Flight Booking CS; FCUL, FCT/UNL, UL and UY are the different groups/institutions)

	SS	Is	Os
VCL	22	23	26
UML	15	11	12
NP	6	9	5
Total	43	43	43

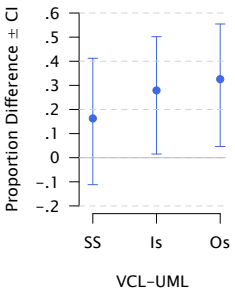
(a) Frequencies



(b) Frequency distribution



(c) Proportions and CIs



(d) Proportion Difference and CIs

Figure 4.3: Notation that users felt gave them a better performance in the model construction tasks as obtained from responses to debriefing questionnaire. (SS = State Space; Is = Invariants; Os = Operations; NP = No preferance)

not significant to reject the null hypothesis (Z-test p-value = .064); Cohen's h effect-size measure yields .33 (CI [.28, .38], •+), signals a small to medium effect size.

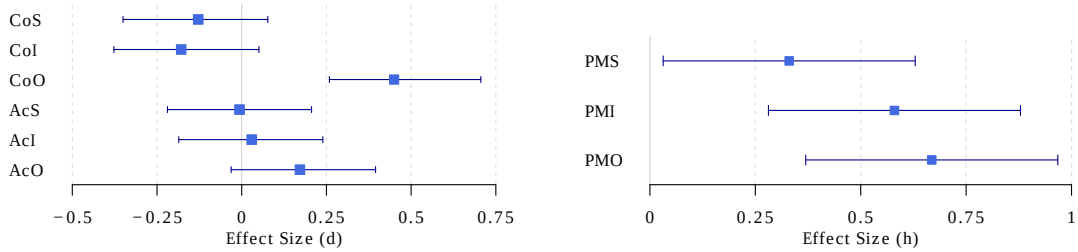
- In modelling the invariants, most subjects felt that they performed better using VCL (23/43 = .53, CI [.39, .67]), followed by $UML + OCL$ (11/43 = .26, CI[.15, .4]) and the *no preference* option (9). The difference of the proportions between VCL and UML+OCL is .28 (CI [.02, .5]). The Z-test analysis rejected the null hypothesis (p-value = .004 < .01, **), hence we accept the alternative hypothesis: $PMI(VCL) > PMI(UML+OCL)$. Cohen's h effect-size measure of .58 (CI [.53, .63], ••) indicates a medium effect.

- In operation contracts, most subjects felt that they performed better using VCL (26/43 = .6, CI[.46, .74]), which outperformed $UML + OCL$ (12/43 = .28, CI[.17, .43]) and the *no preference* option (5). The difference between proportions is: .33 (CI [.05, .55]), which is sufficiently significant to reject the null hypothesis (Z-test p-value = .001 < .01, **), hence: $PMO(VCL) > PMO(UML+OCL)$. Cohen's h effect-size measure of .67 (CI[.62, .71], ••+) signals a medium to large effect.

4.3.4 Discussion

UML+OCL's slight advantage in completeness of state space (variable CoS) could be due to the fact that UML class diagrams are widely known and many participants already knew this notation. However, although subjects modelled more with UML class diagrams, this advantage was cancelled out when it came to accuracy, which measures quality. There is no difference in accuracy between the two approaches. This greater familiarity with the UML for modelling the state space can also be observed in the amplitude of the box plots for both completeness and accuracy. UML box plots are more compact; the higher amplitude of the VCL box plots seems to suggest that some users excelled with it, but others were perhaps perplexed with the new notation and had lower performances.

Completeness of modelling invariants is slightly higher for $UML+OCL$. This can be attributed to the fact that it is easier to write complex properties textually than graphically, as was observed in our published results on tool evaluation [AG15]. It is interesting that both completeness and



(a) Actual modelling performances (effect size d)

(b) Perceived modelling performances (effect size h)

Figure 4.4: Forest Plots of effect sizes and their CIs derived from actual and perceived performances in the model construction tasks.

accuracy were higher for the FCT/UNL group, which could be due to prior exposure of many members of this group to OCL. Like in state space, there is a slight advantage to UML+OCL with completeness, which is cancelled out when it comes to accuracy.

VCL was superior in the modelling of operations. This is, however, more clear with completeness – subjects could model more behaviour with VCL –, which is nearly cancelled out when it comes to accuracy. The difference between the two approaches, as measured by the p-values, goes from highly significant for completeness to just marginally significant for accuracy; the effect size, as measured by the Cohen's d index, goes from medium to low significance.

A striking aspect of the results presented above is the significant gap that exists between subjects' actual and perceived performances. This effect happens for both completeness (sect. 4.3.1) and accuracy (sect. 4.3.2), which are both at odds with what subjects perceived (sect 4.3.3). Overall, most subjects felt that they performed better using VCL at modelling the state space, invariants and contracts, and better using $UML + OCL$ in the definition of operation signatures. However, their actual performances do not reflect this perceived image. In terms of state spaces and invariants, the performances using $UML + OCL$ outperformed VCL, albeit with no significant difference in neither completeness nor accuracy. In the modelling of operations, it was using VCL that subjects performed better, but only with a high significant difference on completeness – much less significant for accuracy.

It is in the modelling of invariants that the gap between actual and perceived performances is particularly evident. In the questionnaire, subjects perceived their performances with VCL as being significantly better than UML+OCL; however, it wasn't so when it came to the actual performances. This reflects, on the one hand, the emotional nature of more subjective opinions – as it will become evident later, subjects favoured VCL –, but on the other hand it suggests the inadequacy of the experiment at measuring the performances of subjects at modelling invariants. Modelling of invariants was done in the same task as modelling the state space with 1/2 hour to do both.

The results of both completeness and accuracy highlighted a striking difference in the performances of modelling the state space when compared to the modelling of invariants and operations. This reflects the difference in complexity of the tasks: state space modelling is easier than the more advanced modelling of invariants and operations. Students and learners of software modelling, first master state space modelling and only then they move to the more advanced topics of modelling invariants and operations. The outliers in the modelling of state space in Fig. 4.1a signal subjects that were not prepared for the complex modelling tasks of the experiment, whereas

the outliers in the modelling of invariants and operations (Figs. 4.1d and 4.1g) signal talented and competent individuals performing well above the average.

As commented above, the results for completeness tend to be higher than those of accuracy. It is much easier to get points for completeness (how much has been modelled) than for accuracy, which has to do with the meaning of what has been modelled.

4.4 Comprehension and Model Usage

The comprehension and model usage tasks comprised problem comprehension, defect detection and model comprehension. We have measured both actual and perceived performance of subjects in these tasks.

4.4.1 Task Performance

The results of the actual performance of subjects in comprehension and model usage tasks are summarised in table 4.4 and portrayed in Fig. 4.5; they correspond to hypothesis $H_{10..12}$. The results are as follows:

- Figures 4.5a, 4.5b and 4.5c, which depict the problem comprehension results, show that the performances of subjects using both approaches is very similar. There is an advantage in favour of VCL for the university library case study, which is cancelled out in the flight booking case study. The results confirm the null hypothesis $PC(VCL) = PC(UML+OCL)$ (Wilcoxon test p-value = .33; t-test p-value =.38). The mean of VCL's performances was .65 (sd= .19, CI $[.61, .69]$) and that of UML+OCL was .64 (sd= .2, CI $[.6, .68]$) with a mean difference of about .01 (CI$[-.04, .05]$). This quasi-equality is endorsed by Cohen's d ES measurement of .03 (CI$[-.18, .24]$, ø), which signals a null effect.

- The results of defect detection, depicted in Figs. 4.5d, 4.5e and 4.5f, highlight a significant difference in favour of VCL – conspicuous in the university library case study. The difference is highly significant, confirming the alternative hypothesis $DD(VCL) > DD(UML)$ (Wilcoxon test p-value = $2 \times 10^{-7} < 0.001$ (***); t-test p-value =3×10^{-8}). The means are as follows: VCL was .27 (sd= .13, CI $[.24, .3]$), UML+OCL was .19 (sd = .1, CI $[.17, .21]$) and the difference was .08 (CI $[.05, .1]$). Cohen's d ES measurement of .67 (CI $[.41, .87]$, ••+) signals a medium to large effect.

- In model comprehension, whose results are portrayed in Figs. 4.5g, 4.5h and 4.5h, there is a significant difference in favour of VCL (Wilcoxon test p-value =0.001 < 0.01 (**); t-test p-value =.003). VCL had a mean score rate of .67 (sd = .16, CI$[.64, .71]$) and UML+OCL of .61 (sd = .16, CI$[.58, .64]$) with a mean difference of .06 (CI $[.02, .1]$). Cohen's d ES measurement of .37 (CI $[.09, .52]$, •+) signals a small to medium effect.

4.4.2 Perceived Performance

The debriefing questionnaire enquired subjects on their performances in the model usage and comprehension tasks. The results of this subjective assessment of user performances, corresponding to hypothesis $H_{13..15}$, are described in Fig. 4.6, which contains a table of frequencies (Fig. 4.6a), a bar chart depicting the frequency distributions (Fig. 4.6), a plot of proportions and CIs (Fig. 4.6c) and a plot of differences between proportions with CIs (Fig. 4.6d). The results are as follows:

	University Library							Flight Booking							Total						
	N	VCL		UML		PD		N	VCL		UML		PD		N	VCL		UML		PD	
		M	S	M	S	M	S		M	S	M	S	M	S		M	S	M	S	M	S
(1)	16	.71[.6, .81]	.21	.68[.6, .75]	.16	.03[−.08, .14]	.22	16	.49[.41, .57]	.17	.55[.46, .64]	.18	−.06[−.13, .01]	.14	32	.6[.52, .68]	.22	.61[.55, .68]	.18	−.02[−.08, .05]	.19
(2)	11	.7[.58, .83]	.22	.69[.59, .8]	.18	.01[−.07, .09]	.14	11	.55[.47, .62]	.13	.63[.46, .79]	.28	−.08[−.25, .08]	.28	22	.62[.55, .7]	.19	.66[.57, .76]	.23	−.04[−.13, .06]	.22
(3)	9	.77[.67, .87]	.15	.58[.44, .71]	.21	.19[.11, .28]	.13	9	.64[.55, .74]	.15	.61[.45, .76]	.23	.04[−.06, .13]	.15	18	.71[.63, .78]	.16	.59[.49, .69]	.22	.11[.04, .19]	.16
(4)	7	.7[.6, .81]	.14	.8[.7, .9]	.13	−.1[−.21, .01]	.15	7	.73[.64, .82]	.12	.67[.54, .8]	.18	.07[−.11, .24]	.24	14	.72[.65, .79]	.13	.73[.65, .82]	.17	−.01[−.12, .09]	.21
(5)	43	.72[.66, .78]	.19	.68[.63, .73]	.18	.04[−.02, .1]	.19	43	.58[.53, .63]	.17	.6[.54, .67]	.22	−.03[−.09, .04]	.2	86	.65[.61, .69]	.19	.64[.6, .68]	.2	.01[−.04, .05]	.2

(a) Problem comprehension (variable *PC*)

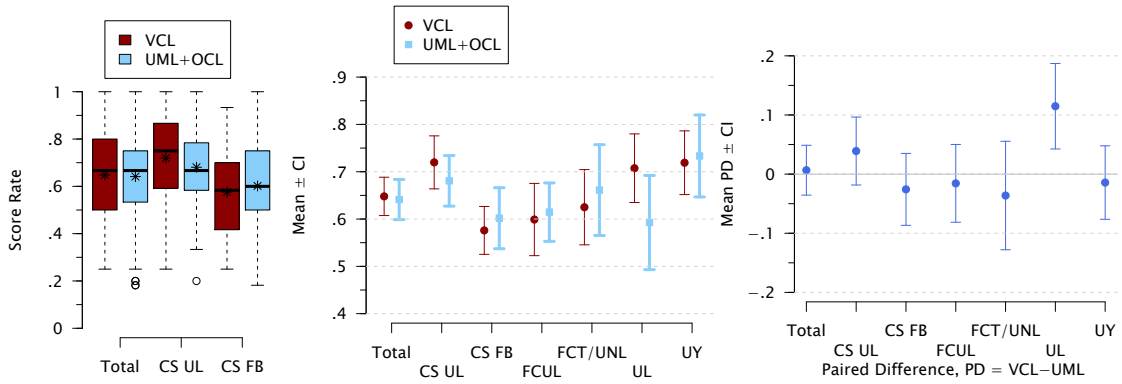
	University Library							Flight Booking							Total						
	N	VCL		UML		PD		N	VCL		UML		PD		N	VCL		UML		PD	
		M	S	M	S	M	S		M	S	M	S	M	S		M	S	M	S	M	S
(1)	16	.29[.21, .36]	.16	.14[.1, .19]	.09	.15[.08, .21]	.12	16	.24[.18, .3]	.12	.19[.15, .24]	.1	.05[.01, .09]	.08	32	.26[.21, .31]	.14	.17[.13, .2]	.1	.1[.06, .14]	.12
(2)	11	.38[.28, .47]	.16	.25[.2, .29]	.08	.13[.04, .22]	.15	11	.23[.17, .3]	.11	.28[.19, .37]	.14	−.05[−.11, .02]	.1	22	.3[.24, .37]	.15	.26[.22, .31]	.12	.04[−.02, .11]	.16
(3)	9	.27[.19, .36]	.13	.13[.07, .18]	.08	.15[.08, .21]	.1	9	.22[.17, .27]	.08	.17[.11, .23]	.09	.05[.01, .1]	.07	18	.25[.2, .3]	.11	.15[.11, .19]	.09	.1[.05, .14]	.1
(4)	7	.3[.23, .37]	.09	.18[.13, .23]	.06	.12[.04, .2]	.11	7	.21[.18, .23]	.03	.18[.14, .23]	.06	.03[−.04, .09]	.09	14	.25[.21, .3]	.08	.18[.15, .21]	.06	.07[.01, .13]	.11
(5)	43	.31[.27, .35]	.14	.17[.14, .2]	.09	.14[.1, .17]	.12	43	.23[.2, .26]	.1	.21[.17, .24]	.11	.02[−.01, .05]	.09	86	.27[.24, .3]	.13	.19[.17, .21]	.1	.08[.05, .1]	.12

(b) Defect detection (variable *DD*)

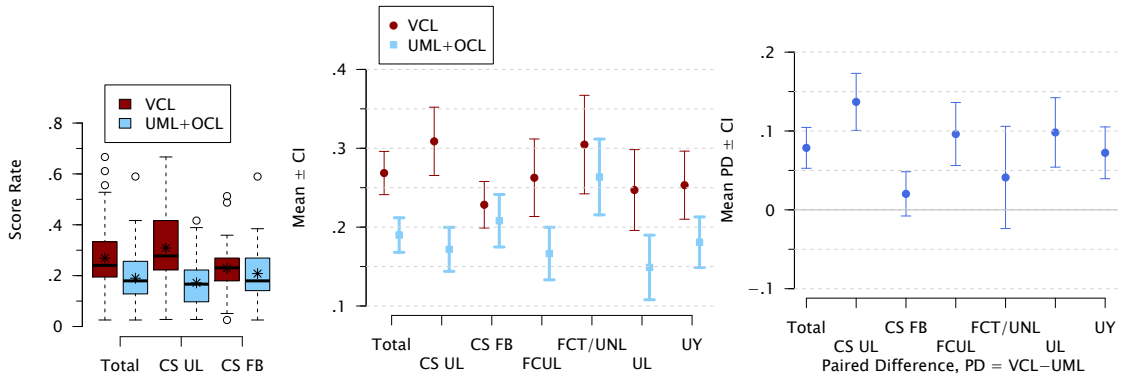
	University Library							Flight Booking							Total						
	N	VCL		UML		PD		N	VCL		UML		PD		N	VCL		UML		PD	
		M	S	M	S	M	S		M	S	M	S	M	S		M	S	M	S	M	S
(1)	16	.67[.6, .75]	.15	.58[.48, .67]	.2	.1[.001, .19]	.2	16	.69[.64, .75]	.11	.67[.6, .74]	.14	.02[−.06, .1]	.17	32	.68[.64, .73]	.13	.62[.56, .69]	.18	.06[−.004, .12]	.18
(2)	11	.36[.28, .45]	.15	.24[.19, .29]	.09	.12[.04, .21]	.15	11	.22[.16, .28]	.1	.27[.19, .35]	.14	−.04[−.1, .02]	.1	22	.72[.67, .77]	.13	.61[.54, .68]	.17	.11[.04, .19]	.17
(3)	9	.45[.29, .62]	.26	.58[.49, .66]	.13	−.13[−.31, .05]	.27	9	.69[.64, .75]	.08	.56[.45, .66]	.17	.14[.05, .23]	.14	18	.57[.47, .68]	.22	.57[.5, .63]	.15	.01[−.11, .12]	.25
(4)	7	.64[.52, .77]	.17	.66[.55, .78]	.16	−.02[−.14, .1]	.16	7	.73[.61, .86]	.17	.62[.53, .71]	.12	.11[−.05, .27]	.21	14	.69[.6, .78]	.17	.64[.57, .71]	.14	.05[−.06, .15]	.19
(5)	43	.64[.58, .7]	.2	.6[.55, .65]	.17	.04[−.03, .11]	.23	43	.7[.67, .74]	.12	.62[.58, .67]	.15	.08[.03, .13]	.16	86	.67[.64, .71]	.16	.61[.58, .64]	.16	.06[.02, .1]	.2

(c) Model comprehension (variable *MC*)

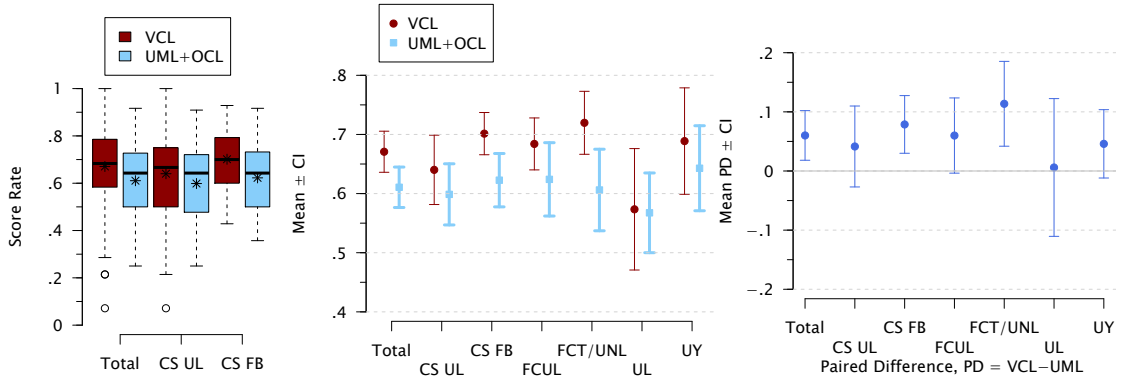
Table 4.4: Experimental data on comprehension and model usage. ((1) = FCUL; (2) = FCT/UNL; (3) = U. Luxembourg; (4) = U. York; (5) = Total)



(a) Box plots of problem comprehension (b) Means and CIs for problem comprehension (c) Paired differences and CIs of problem comprehension



(d) Box plots of defect detection (e) Means and CIs for defect detection (f) Paired differences and CIs of defect detection

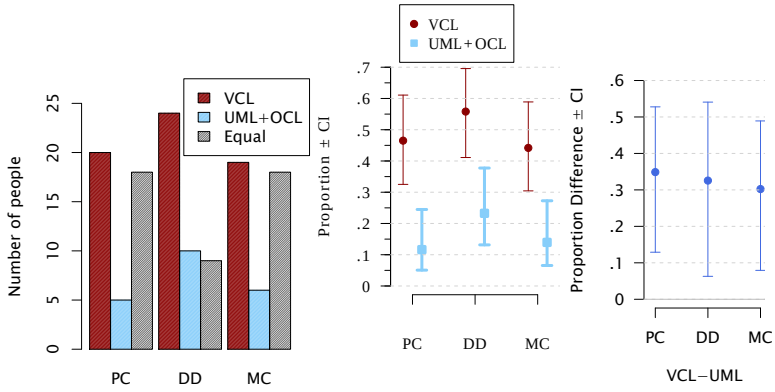


(g) Box plots of model comprehension (h) Means and CIs for model comprehension (i) Paired differences and CIs of model comprehension

Figure 4.5: Model usage and comprehension tasks (CS UL = University Library case study (CS); CS FB = Flight Booking CS; FCUL, FCT/UNL, UL and UY are the different groups/institutions).

	PC	DD	MC
VCL	24	24	19
UML	5	10	6
Equal	18	9	18
Total	43	43	43

(a) Frequencies



(b) Histogram of frequency distributions

(c) Proportions and CIs

(d) Proportion differences and CIs

Figure 4.6: Notation that users felt gave them a better performance in the model usage and comprehension tasks: problem comprehension, defect detection and model comprehension according to responses to debriefing questionnaire (PC = Problem Comprehension; DD = Defect Detection; MC = Model Comprehension)

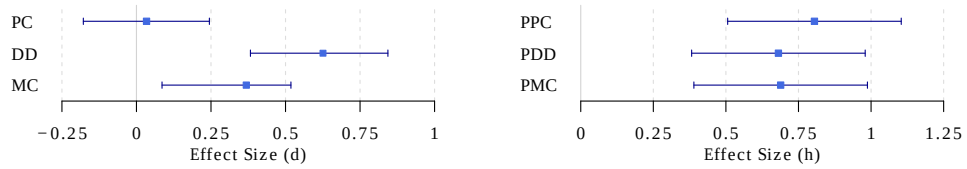
- In problem comprehension (PC), a proportion of .47 (CI[.33, .61]) of subjects felt that they performed better using VCL, against .12 (CI [.05, .24]) of UML+OCL, yielding a difference .35 (CI [.13, .53]). As can be inferred from Figs. 4.6c, this difference is notably significant (Z-test p-value=.0002 $<$ .001, ***), hence, we reject the null hypothesis and accept the alternative hypothesis: $PPC(VCL) > PPC(UML + OCL)$. Cohen's h effect-size measure of .8 (CI [.51, 1.1], ●●●) signals a large effect.

- In defect detection (DD), a considerable proportion of subjects, .56 (CI [.41, .7]) felt they performed better using VCL against UML+OCL's .23 (CI [.13, .38]) – the difference is .33 (CI [.06, .54]). Figures 4.6c and 4.6d highlight a significant difference in favour of VCL confirmed by our Z-test p-value of .001 $<$ .01 (**), which rejects the null and accepts the alternative hypothesis: $PDD(VCL) > PDD(UML + OCL)$. Cohen's h effect-size measure of .68 (CI [.38, .98], ●●+) signals a medium to large effect.

- In model comprehension (MC), a considerable proportion of subjects, .44 (CI [.3, .59]), felt they performed better using VCL against UML+OCL's .14 (CI [.07, .27]) – the difference is .3 (CI [.08, .49]). Figures 4.6c and 4.6d highlight a significant difference in favour of VCL confirmed by our Z-test p-value of .001 $<$ .01 (**), entailing the acceptance of the alternative hypothesis: $PMC(VCL) > PMC(UML + OCL)$. Cohen's h effect-size measure of .69 (CI [.39, .99], ●●+) indicates a medium to large effect.

4.4.3 Summary

Figure 4.7 presents a forest plot describing actual and perceived performances in the model usage and comprehension tasks. It depicts the standardised difference in means effect size (Cohen's d) for each relevant hypothesis together with the CIs. There is an interesting contrast between actual and perceived performances. We can see that for defect detection and model comprehension, actual and perceived are not that far apart; the corresponding null hypothesis were



(a) Actual performances (effect size d) (b) Perceived performances (effect size h)

Figure 4.7: Forest Plots of effect sizes of actual and perceived performances in the model usage and comprehension tasks.

Table 4.5 Experimental data on Usefulness and ease of use. ((1) = FCUL; (2) = FCT/UNL; (3) = U. Luxembourg; (4) = U. York; (5) = Total)

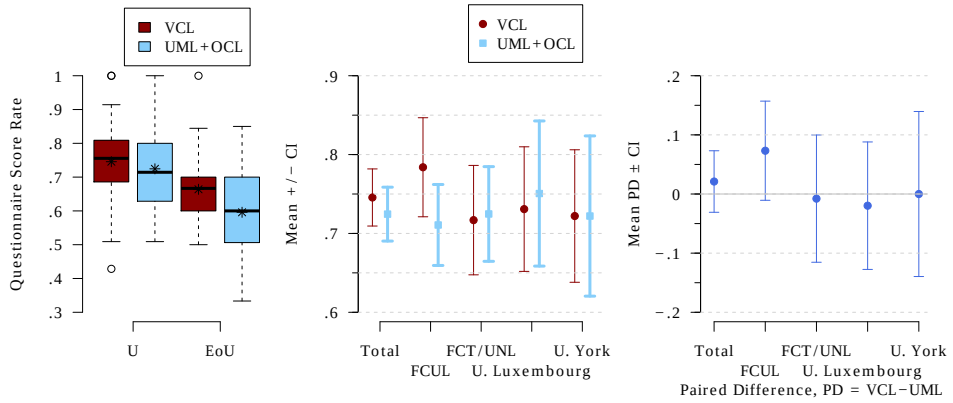
	Usefulness (U)							Ease of Use (EoU)						
	N	VCL		UML		PD		N	VCL		UML		PD	
		M	S	M	S	M	S		M	S	M	S	M	S
(1)	16	.78[.72, .85]	.13	.71[.66, .76]	.1	.07[−.01, .16]	.17	16	.68[.62, .73]	.11	.6[.55, .65]	.1	.08[.02, .14]	.12
(2)	11	.72[.65, .79]	.12	.72[.66, .78]	.1	−.01[−.12, .1]	.18	11	.66[.61, .72]	.09	.59[.51, .66]	.12	.08[−.01, .16]	.14
(3)	9	.73[.65, .81]	.12	.75[.66, .84]	.14	−.02[−.13, .09]	.16	9	.66[.59, .73]	.11	.62[.54, .71]	.13	.03[−.08, .15]	.17
(4)	7	.72[.64, .81]	.11	.72[.62, .82]	.14	0[−.14, .14]	.19	7	.65[.58, .72]	.09	.57[.48, .67]	.13	.08[−.06, .22]	.19
(5)	43	.75[.71, .78]	.12	.72[.69, .76]	.11	.02[−.03, .07]	.17	43	.66[.63, .69]	.1	.6[.56, .63]	.11	.07[.03, .11]	.14

rejected. There is, however, a significant discrepancy in the the way subjects perceive problem comprehension and their actual performance in the problem comprehension questionnaires. Subject's VCL perceived performance in problem comprehension was significantly better than the *UML + OCL* one with a very strong effect size, however there is a close to null difference in the actual performance. This suggests a possible weakness of the experiment, which way not have been effective in evaluating problem comprehension to the point that it could not identify a difference in terms of performance, although the subjects felt that there was a difference; however, it can also be that the perceived difference is illusional.

4.5 Usefulness and Ease of Use

The experiment's debriefing questionnaire contained sections devoted to *perceived usefulness* (PU) and *perceived ease of use* (PEoU) [Dav89]. Perceived usefulness is the degree to which a person believes that using a particular system would enhance his or her job performance. Perceived ease of use, on the other hand, is the degree to which a person believes that using a particular system or technology would be free of effort. Both perceived usefulness and perceived ease of use are seen as important determinants for user acceptance of a technology [Dav89].

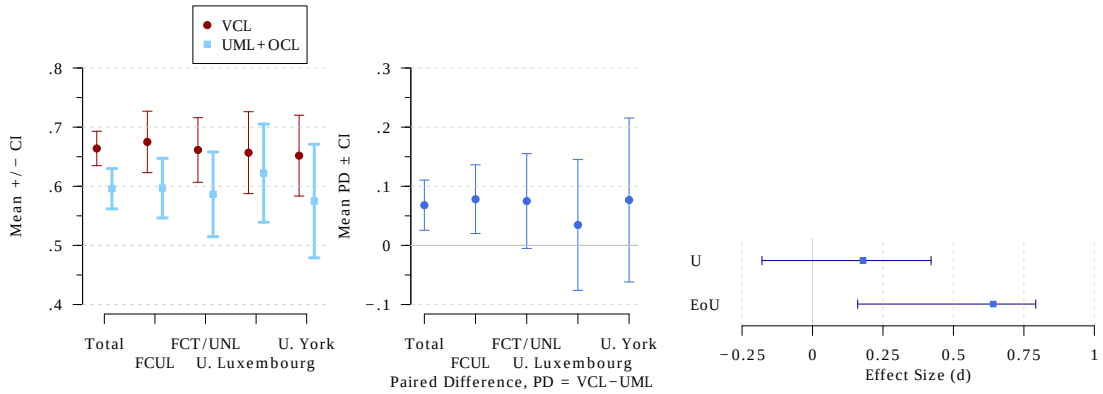
To evaluate PU and PEoU, the questionnaire contained indirect and direct questions. The indirect parts of the questionnaire are based on questions formulated on a Likert scale from 1 (strongly agree) to 5 (strongly disagree), corresponding to variables U and EoU and the hypothesis $H_{16,17}$. The results, summarised in table 4.5, are portrayed in the plots of Fig. 4.8, which contains a box-plot describing the distributions of U and EoU (Fig. 4.8a), a plot of mean and CIs of U (Fig. 4.8b), a plot of mean differences and their CIs for U (Fig. 4.8c), a plot of mean and CIs of EoU (Fig. 4.8d), a plot of mean differences and their CIs for EoU (Fig. 4.8e) and a



(a) Box plots (U and EoU)

(b) Means and CIs (U)

(c) Paired mean differences and CIs (U)



(d) Means and CIs (EoU)

(e) Means and CIs of paired differences and CIs (EoU)

(f) Forest plot of effect sizes (d)

Figure 4.8: Distributions of usefulness and ease of use (indirect questions)

forest-plot of effect sizes (Fig. 4.8f). The results are as follows:

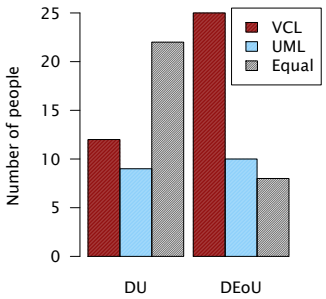
- The usefulness results give a slight advantage to VCL, which is not statistically significant (Wilcoxon test p-value =.095; t-test p-value=.22) to confirm the null hypothesis. The mean of VCL's usefulness survey is .75 (sd=.12, CI [.71, .78]) and that of UML+OCL is .72 (sd=.11, CI [.69, .76]) with a mean of differences of .02 (sd=.17, CI $[-0.03, 0.07]$). Cohen's d effect size measurement of .18 (sd=.14, CI $[-.18, .42]$, •) signals a small effect.

- In ease of use, on the other hand, there was a significant advantage in favour of VCL — the Wilcoxon test p-value is .003 < .01 (**) (t-test p-value = .002). The mean of VCL's PEoU survey is .66 (sd=.1, CI [.63, .69]), and that of UML+OCL is .6 (sd=.11, CI [.56, .63]) with a mean of differences of .07 (sd=.14, CI [.03, .11]). Cohen's d effect size measurement of .64 (CI [.16, .79], ••+) signals a medium to large effect.

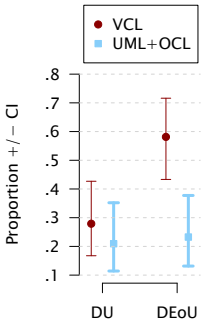
The results to the direct questions on PU and PEoU, corresponding to hypothesis $H_{18,19}$, are

	DU	DEoU
	•	*** ●●+
VCL	12	25
UML	9	10
Equal	22	8
Total	43	43

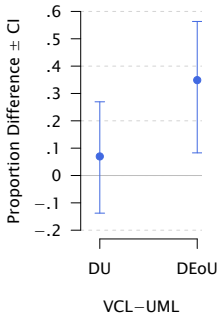
(a) Frequencies, significances and effects



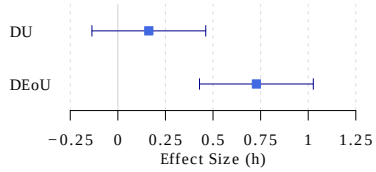
(b) Bar chart of frequency distributions



(c) Means and CIs



(d) Proportions and CIs of paired differences



(e) Forest plot of effect sizes (d)

Figure 4.9: Distributions of usefulness and ease of use in the direct questions (variables: DU and DEoU)

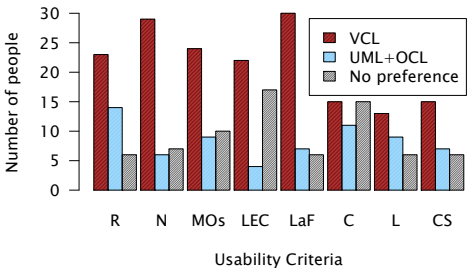
described in Fig. 4.9, which contains a table of frequencies (table 4.11a), a bar chart depicting the frequency distributions (Fig. 4.9b), a plot of proportions and CIs (Fig. 4.9c), a plot of proportion differences and CIs (Fig. 4.9d) and a forest plot of effect sizes (Fig. 4.9e). The results are as follows:

- The usefulness frequencies show that most users selected the neutral option of the questionnaire (22 out of 43), followed by VCL (12) and $UML + OCL$ (9). This gives a proportion of .28 (CI $[.17, .43]$) to VCL and a proportion of .21 (CI$[.11, .35]$) to $UML + OCL$ with a difference in proportions of .07 (CI $[-.14, .27]$). This difference is not statistically significant (Z-test p-value=.23) and confirms the null hypothesis. Cohen's h effect-size measure of .16 (CI $[-.14, .46]$, •) signals a small effect.

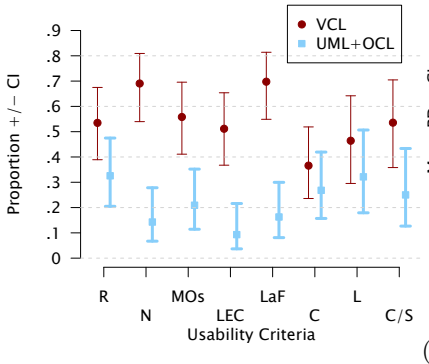
- The ease of use results show that most users selected VCL as the language that is more easy to use (25 out of 43), followed by $UML+OCL$ (10) and the neutral option (8). This gives a proportion of .58 (CI $[.43, .72]$) to VCL against the .23 (CI $[.13, .38]$) of $UML+OCL$, yielding a difference of .35 (CI $[.08, .56]$). This difference was deemed statistically significant; the Z-test p-value of .0005 ($< .001$, ***) confirms the alternative hypothesis: $DEoU(VCL) > DEoU(UML + OCL)$. Cohen's h effect-size measure of .73 (CI $[.43, 1.03]$, ●●+) signals a medium to large effect.

	R	N	MOs	LEC	LaF	C	L	C/S
	*	***	***	***	***			*
	••	•••	••	•••	•••	•	•	••
			+			+	+	
VCL	23	29	24	22	30	15	13	15
UML	14	6	9	4	7	11	9	7
NP	6	7	10	17	6	15	6	6
Total	43	42	43	43	43	41	28	28

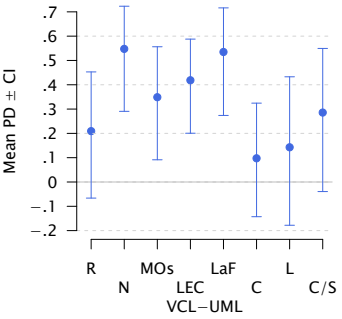
(a) Frequencies, significances and effects



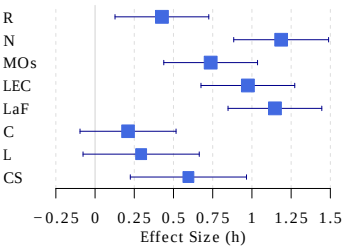
(b) Frequency distributions



(c) Proportions and CIs



(d) Proportion differences and CIs



(e) Forest plot of effect sizes (h)

Figure 4.10: Usability results derived from responses to the debriefing questionnaire. (NP= No preference. Criteria: R = Reading; N = Navigation; MOs = Maps and overviews; LEC = live error checking; LaF = Look and feel; C=Cohesion; L=Learnability; CS=Comfort Satisfaction. Significance: * corresponds to $\alpha = .05$, ** to $\alpha = .01$, and *** to $\alpha = .001$. Effect magnitude of h: ø is null ($|\ h\ | \leq .05$), • is small ($.05 < |\ h\ | \leq .2$), •+ is small to medium ($.2 < |\ h\ | < .4$), •• is medium ($.4 \leq |\ h\ | \leq .6$), ••+ is medium to large ($.6 < |\ h\ | < .8$), ••• is large ($|\ h\ | \geq .8$).)

4.6 Usability

The usability section of the debriefing questionnaire comprised closed questions on several usability criteria with the alternatives *VCL*, *UML+OCL* and *no preference*. The results, corresponding to hypothesis $H_{20..27}$, are portrayed in Fig 4.10, which contains a table of frequencies (Fig. 4.11a), a bar chart depicting the frequency distributions (Fig. 4.10b), a plot of proportions and CIs (Fig. 4.10c), a plot of proportion differences and CIs (Fig. 4.10d), and a forest plot of effect sizes (Fig. 4.10e). VCL was the most voted option in every selected criterium, as can be inferred from the table in Fig. 4.11a and the bar chart in Fig. 4.10b. The results are as follows:

- In reading (R), VCL obtained a proportion of .53 (CI $[.39, .67]$) against .33 (CI $[.19, .47]$) of $UML + OCL$, yielding a difference of .21 (CI $[-.07, .45]$). This difference was deemed significant by the Z test p-value of .025 ($< .05$, *) — hence, $UsR(VCL) > UsR(UML + OCL)$. The ES measurement Cohen's h of .43 (CI $[.13, .73]$, ••) signals a medium effect.
- In navigation (N), VCL obtained a proportion of .69 (CI $[.54, .81]$) against $UML + OCL$'s .14 (CI $[.07, .28]$), yielding a difference of .55 (CI $[.29, .72]$), which is highly significant — the Z test p-value is 2×10^{-7} ($< .001$, ***). Cohen's h ES yields 1.19 (CI $[.88, 1.49]$, •••), a comfortably large effect.

- In maps and overviews (MOs), VCL obtained a proportion of .56 (CI [.41, .7]) against .21 (CI [.11, .35]) of *UML + OCL*. The difference of .35 (CI [.09, .56]) is highly significant — the Z test p-value is .0004 (< .001, ***). Cohen's h ES of .74 (CI [.31, 1.16], ••+) denotes a medium to large effect.

- In live error checking (LEC), VCL was selected by a .51 (CI [.37, .65]) proportion of the subjects against a proportion of .09 (CI [.04, .22]) for UML+OCL, yielding a difference of .42 (CI [.2, .59]). This difference is highly significant – the Z test p-value is 10^{-5} (< .001, ***). Cohen's h value of .97 (CI [.68, 1.27], •••) indicates a large effect.

- In look and feel (LaF), VCL obtained a proportion of .7 (CI [.55, .81]) against UML+OCL's .16 (CI [.08, .3]) for UML+OCL, with a difference of .53 (CI [.27, .72]). This difference was deemed highly significant – the Z test p-value yields 3×10^{-7} (< .001, ***). ES Cohen's h of 1.15 (CI [.85, 1.45], •••) denotes a comfortably large effect.

- In cohesion (C), VCL obtained a proportion of .37 (CI [.24, .52]) against .27 (CI [.16, .42]) of *UML + OCL*, yielding a difference of .1 (CI [−.14, .32]), which is not significant (Z-test p-value = .17). ES Cohen's h of .21 (CI [−.1, .52], •+) signals a small to medium effect.

- In learnability (L), VCL obtained a proportion of .46 (CI [.3, .64]) against .32 (CI [.18, .51]) of *UML + OCL*, yielding a difference of .14 (CI [−.18, .43]) that is not significant (Z-test p-value = .14). The ES measurement Cohen's h is .29 (CI [−.23, .82], • +), a small to medium effect.

- In comfort/satisfaction (C/S), VCL got proportion of .54 (CI [.36, .7]) and UML+OCL of .25 (CI [.13, .43]). This yields a difference of .29 (CI [−.04, .55]), which is significant (Z test p-value = .014 < .05, *). Cohen's h ES measurement of .6 (CI [.22, .97], ••) denotes a medium effect.

4.7 Preferred Notation

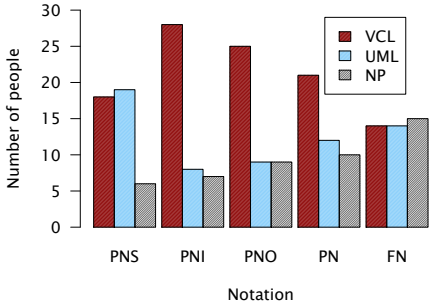
The preferred notation section of the debriefing questionnaire enquired subjects on their preferred notation based on their experience of using both notations to carry out the tasks of the experiment. This comprised closed questions with the alternatives *VCL*, *UML+OCL* and *no preference*, which correspond to the hypothesis $H_{28..32}$. The results are portrayed in Fig. 4.11, which contains a table of frequencies with ratings of significance and ratings of effect size (table in Fig. 4.11a), a bar chart picturing the frequency distributions of the relevant dependant variables (Fig. 4.11b), a plot of means and CIs (Fig. 4.11c), and a forest plot depicting the effect sizes (Fig. 4.11e). The results are as follows:

- In the preferred notation for the state space (PNS), there is no significant difference between VCL ($P = .42$, CI[.28, .57]) and UML+OCL ($P = .44$, CI [.3, .59]). The difference of −.02 (CI [−.29, .24]) is far from being significative (Z-test p-value = .59). The measured effect size is null ($h = -.05$, CI[−.35, .25]).

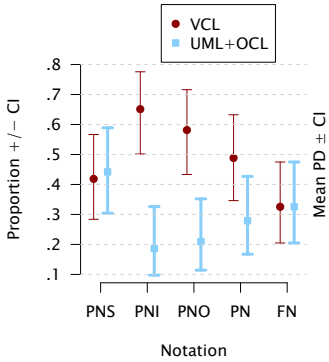
- In the preferred notation for invariants (PNI), there is a highly significant difference in favour of VCL ($P = .65$, CI[.5, .78]) against UML+OCL ($P = .19$, CI [.1, .33]). The difference of .47 (CI [.2, .66]) is very significant (Z-test p-value = $6 \times 10^{-6} < .001$, ***). The effect size measurement ($h = .99$, CI[.56, 1.41], •••) denotes a large effect.

	PNS	**PNI**	**PNO**	**PN**	**NF**
		***	***	*	
	ø	●●●	●●+	●●	ø
VCL	18	28	25	21	14
UML	19	8	9	12	14
NP	6	7	9	10	15
Total	43	43	43	43	43

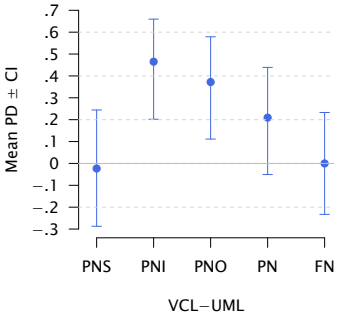
(a) Frequencies, significances and effects



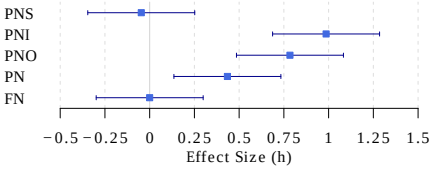
(b) Frequency distributions



(c) Proportions and CIs



(d) Proportion differences and CIs



(e) Forest plot of effect sizes (h)

Figure 4.11: Results of the preferred notation section in the debriefing questionnaire. (NP = No preference. Variables: PNS = Preferred Notation for State Space; PNI = Preferred Notation for Invariants; PNO = Preferred Notation for Operations; PN = Preferred Notation overall; FN = Notation to be used in the Future. Significance: * corresponds to $\alpha = .05$, ** to $\alpha = .01$, and *** to $\alpha = .001$. Effect magnitude of h: ø is null ($|\ h\ | \leq .05$), • is small ($.05 < |\ h\ | \leq .2$), •+ is small to medium ($.2 < |\ h\ | < .4$), •• is medium ($.4 \leq |\ h\ | \leq .6$), ••+ is medium to large ($.6 < |\ h\ | < .8$), ••• is large ($|\ h\ | \geq .8$).)

- Likewise for the preferred notation for operations (PNO). VCL obtains a proportion of .58 (CI $[.43, .73]$) against .21 (CI$[.11, .35]$) of UML+OCL; this gives difference of .37 (CI $[.11, .58]$), which is very significant (Z-test p-value = .0002 $<$ .001, ***). The effect size measurement ($h = .78$, CI$[.49, 1.08]$, $\bullet\bullet+$) signals a medium to large effect.
- In the overall preferred notation (PN), VCL obtains a proportion of .49 (CI $[.35, .63]$) against .28 (CI $[.17, .43]$) of UML-OCL, which gives a difference of .21 (CI $[-.05, .44]$). This is significant (Z-test p-value = .023 $<$ .05, *). The effect size measurement ($h = .43$, CI $[.14, .73]$, $\bullet\bullet$) indicates a medium difference.
- There is a tie in the notation to be used in the future (NF). The proportions of VCL and UML+OCL are the same — .33 (CI $[.2, .47]$) . The difference is null ($= 0$, CI $[-.23, .23]$); likewise for the effect size ($h = 0$, CI $[-.3, .3]$).

Chapter 5

Threats to Validity

This chapter will discuss the different types of threats to Validity of the VCLvsUML+OCL experiment.

References

[ABHL06] Erik Arisholm, Lionel C. Briand, Siw Elisabeth Hove, and Yvan Labiche. The impact of UML documentation on software maintenance: an experimental evaluation. *IEEE TSE*, 32(6):365–381, 2006.

[AG15] Nuno Amálio and Christian Glodt. A tool for visual and formal modelling of software designs. *Science of Computer Programming*, 98, Part 1:52 – 79, 2015.

[AGK11] Nuno Amálio, Christian Glodt, and Pierre Kelsen. Building VCL models and automatically generating Z specifications from them. In *FM 2011*. Springer, 2011.

[AK10] Nuno Amálio and Pierre Kelsen. Modular design by contract visually and formally using VCL. In *VL/HCC 2010*, 2010.

[AKMG10] Nuno Amálio, Pierre Kelsen, Qin Ma, and Christian Glodt. Using VCL as an aspect-oriented approach to requirements modelling. *TAOSD*, VII:151–199, 2010.

[Amá11] Nuno Amálio. VCL model of the secure simple bank case study. Technical Report TR-LASSY-11-05, Univ. of Luxembourg, 2011. `http://bit.ly/q1LrPj`.

[Amá12a] Nuno Amálio. The type system of VCL. Technical Report TR-LASSY-12-10, Univ. of Luxembourg, 2012. `http://bit.ly/gu7wv5`.

[Amá12b] Nuno Amálio. The VCL model of the barbados crisis management system. Technical Report TR-LASSY-12-09, Univ. of Luxembourg, 2012. `http://bit.ly/W5C8ZY`.

[BLPYB05] Lionel C. Briand, Yvan Labiche, Massimiliano Di Penta, and Han Yan-Bondoc. An experimental investigation of formality in UML-based development. *IEEE TSE*, 31(10):833–849, 2005.

[CF01] Geoff Cumming and Sue Finch. A primer on the understanding, use, and calculation of confidence intervals that are based on central and noncentral distributions. *Educational and Psychological Measurement*, 61(4):532–574, 2001.

[Coh88] Jacob Cohen. *Statistical power analysis for the behavioural sciences*. Lawrence Erlbaum Associates, 1988.

[Coh94] Jacob Cohen. The earth is round ($p < .05$). *American Psychologist*, 49(2):997–1003, 1994.

[Cum12] Geoff Cumming. *Understanding the new statistics: Effect sizes, confidence intervals and meta-analysis*. Routledge, 2012.

[Dav89] Fred D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13(3):319–340, 1989.

[Fer77] Eugene S. Ferguson. The mind's eye: nonverbal thought in technology. *Science*, 197(4306), 1977.

[Fer92] Eugene S. Ferguson. *Engineering and the Mind's eye*. MIT Press, 1992.

[FL12] Volker H. Franz and Geoffrey R. Loftus. standard errors and confidence intervals in within-subjects designs: Generalizing loftus and masson (1994) and avoiding the biases of alternative accounts. *Psychonomic Bulletin & Review*, 19(3):395–404, 2012.

[GK05] Robert J. Grissom and John J. Kim. *Effect sizes for Research: A broad practical approach*. Lawrence Erlbaum Associates, 2005.

[Goo00] P Goolkasian. Pictures, words, and sounds: from which format are we best able to reason? *Journal of General Psychology*, 127(4):439–459, 2000.

[Har87] David Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[Har88] David Harel. On visual formalisms. *Commun. of the ACM*, 31(5):514–530, 1988.

[Kel07] Ken Kelley. Confidence intervals for standardized effect sizes: Theory, application, and implementation. *Journal of Statistical Software*, 20(8):1–24, 2007.

[KM95] Young-Gul Kim and Salvatore T. March. Comparing data formalisms. *CACM*, 38(6):103–115, 1995.

[LA12a] Jérôme Leemans and Nuno Amálio. Modelling a cardiac pacemaker visually and formally. In *VL/HCC 2012*, pages 257–258. IEEE, 2012.

[LA12b] Jérôme Leemans and Nuno Amálio. A VCL model of a cardiac pacemaker. Technical Report TR-LASSY-12-04, University of Luxembourg, 2012. `http://bit.ly/xiob5d`.

[Lee11] Jérôme Leemans. Model a pacemaker system using VCL. Master's thesis, FSTC, Univ of Luxembourg, 2011.

[LS87] Jill H. Larkin and Herbert A. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11:65–99, 1987.

[Mey92] Bertrand Meyer. Applying "design by contract". *Computer*, 25(10):40–51, 1992.

[Moo02] Daniel L. Moody. Complexity effects on end user understanding of data models: an experimental comparison of large data model representation methods. In *ECIS 2002*, 2002.

[Moo09] Daniel L. Moody. The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE TSE*, 6(35):756–779, 2009.

[MvH08] Daniel Moody and Jos van Hillegersberg. Evaluating the visual syntax of UML: An analysis of the cognitive effectiveness of the UML family of diagrams. In *SLE*, volume 5452 of *LNCS*. Springer, 2008.

[NA00] Robert G. Newcombe and Douglas G. Altman. Proportions and their differences. In *Statistics with Confidence: Confidence Intervals and Statistical Guidelines*. Wiley, 2000.

[New98] Robert G. Newcombe. Two-sided confidence intervals for the single proportion: comparison of seven methods. *Statistics in Medecine*, 17(8):857–72, 1998.

[OMG12] OMG. OMG systems modeling language, version 1.3. Technical report, OMG, 2012.

[PCM+01] Helen C. Purchase, Linda Colpoys, Matthew McGill, David Carrington, and Carol Britton. UML class diagram syntax: an empirical study of comprehension. In *APVis'01*, 2001.

[PCMC02] Helen C. Purchase, Linda Colpoys, Matthew McGill, and David Carrington. UML collaboration diagram syntax: an empirical study of comprehension. In *VISSOFT 2002*, pages 13–22. IEEE, 2002.

[Pin90] Steven Pinker. A theory of graph comprehension. In R. Freedle, editor, *Artificial Inteligence and the future of testing*, pages 73–126. 1990.

[PJ13] Roland Pfister and Markus Janczyk. Confidence intervals for two sample means: Calculation, interpretation, and a few simple rules. *Advances in cognitive Psychology*, 2013.

[R C15] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015.

[Tob12] Eric Tobias. Visual modelling of and on tangible user interfaces. Master's thesis, University of Luxembourg, 2012.

[TRA12a] Eric Tobias, Eric Ras, and Nuno Amálio. Suitability of visual modelling languages for modelling tangible user interface applications. In *VL/HCC 2012*. IEEE, 2012.

[TRA12b] Eric Tobias, Eric Ras, and Nuno Amálio. VML Usability for Modelling TUI Scenarios - A Comparative Study. Technical Report TR-LASSY-12-06, University of Luxembourg, LASSY, 2012. available at `http://vcl.gforge.uni.lu/doc/VMLCaseStudy.pdf`.

[WRH+12] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering*. Springer, 2012.

[Zha97] Jiajie Zhang. The nature of external representations in problem solving. *Cognitive Science*, 21(2):179–217, 1997.

Appendix A

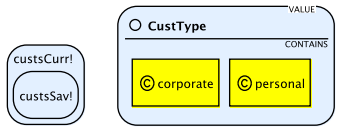
VCL in Nutshell

This appendix gives an overview of VCL using the *secure simple bank* case study [Amá11]. All diagrams presented here have been drawn using the VCB tool. A tutorial on how to build the complete model of this case study in VCB is given in `http://bit.ly/2awqWTq`.

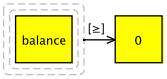
The next sections present VCL's visual primitives and diagram types; each diagram type is illustrated using the case study.

A.1 Visual Primitives

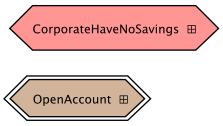
A VCL model is organised around *packages*, represented as *clouds* to allude to a world of their own. VCL packages encapsulate structure and behaviour. In VCL, it is possible to construct larger packages from smaller ones using *incorporation* (or embedding), which is represented in VCL as *enclosure*. In Fig. A.1f, `Authorisation` incorporates the enclosed packages, which means that `Authorisation` is the composition of the incorporated packages plus something else.



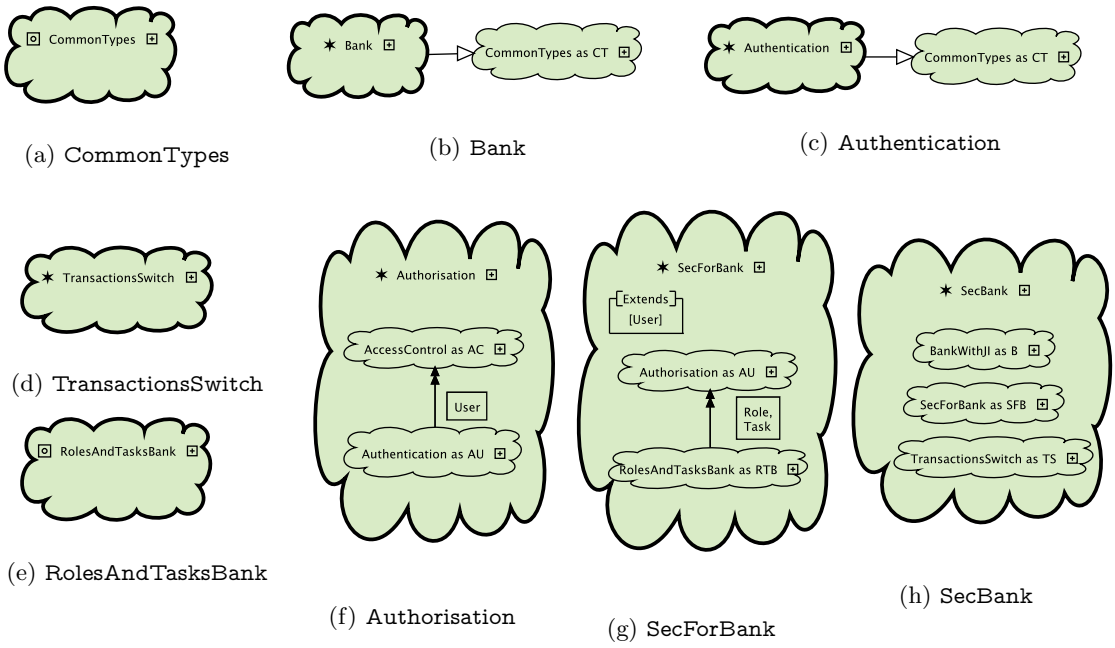
VCL represents *sets* as labelled rounded contours. Topological notion of *enclosure* can either denote the subset relation (to the left, `custsSav!` is subset of `custsCurr!`) or, when accompanied by symbol ◯, define a set in terms of the enclosed sets and objects (to the left, `CustType` is defined by enumerating its elements). *Objects* are represented as rectangles; they denote an element of some set (e.g. `corporate` and `personal` to the left).



Edges describe relations between sets and objects. The *property edge* (labelled arrow) to the left describes a predicate. In Fig. A.2b, the property edges are definitional, introducing properties of sets.



Assertions (labelled hexagons) represent some state constraint or observe operation (a query). They refer to a single system state (e.g. assertion `CorporateHaveNoSavings` to the left). *Contracts* (labelled double-lined hexagons) represent operations that change state; hence, they are double-lined as opposed to single-lined assertions (e.g. `OpenAccount`, left).



(a) CommonTypes (b) Bank (c) Authentication

(d) TransactionsSwitch

(e) RolesAndTasksBank

(f) Authorisation (g) SecForBank (h) SecBank

Figure A.1: Sample package diagrams of VCL model of *secure simple bank*

A.2 VCL Diagram Types

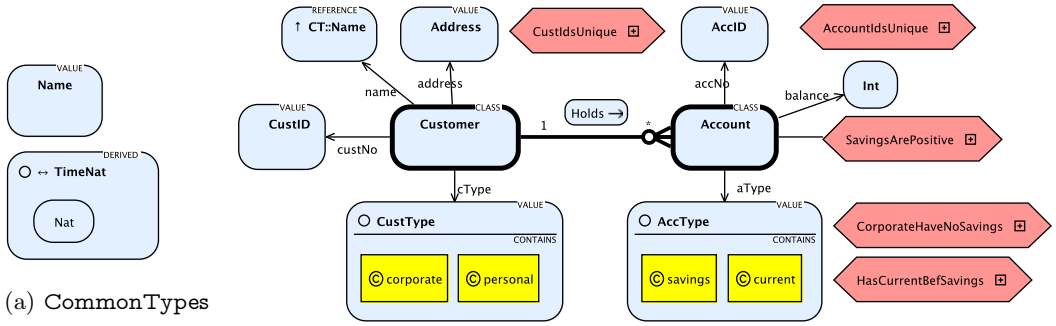
A VCL model comprises a collection of packages. Each package comprises one package Package Diagram (PD), one Structural Diagram (SD), one Behaviour Diagram (BD) and several Assertion Diagrams (ADs) and Contract Diagrams (CDs). The following sections briefly describe each diagram type of VCL.

A.3 Package Diagrams

PDs define VCL packages. They include the package being defined (the current package) and its dependencies with other packages. Figure A.1 presents sample PDs of secure simple bank. PDs are as follows:

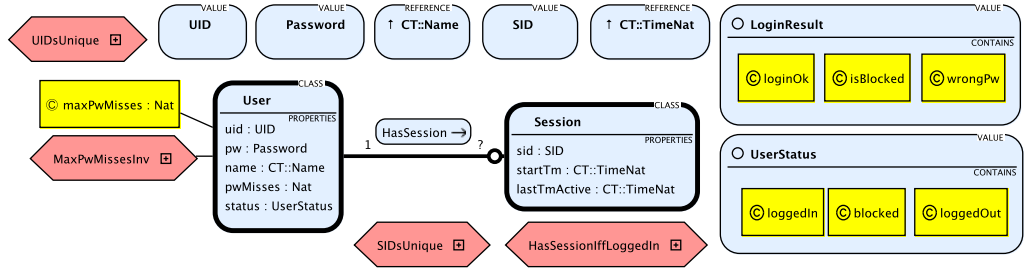
- The package being defined or the *current package* is represented with a bold line. All other packages in a PD are foreign. The symbol ⊞, which can be attached to some package, assertion or contract, has an *expansion* meaning: double-cliking opens the definition. In VCB, double-clicking on packages of a PD gives the user a choice to navigate to the package's package, structure or behaviour diagrams. In Figs. A.1b and A.1h, current packages are, respectively, `Bank` and `SecBank`.

- The current package can be defined as *container* (symbol ⊡) or *ensemble* (symbol ✶). The former act as mere containers for sets and their local operations; containers do not constitute a functional unit, they do not have have a global identity and so they cannot include relation edges and global operations. Ensemble packages, on the other hand, constitute a functional unit on their own, and so they have a global identity, comprising relation edges



(a) CommonTypes

(b) Bank



(c) Authentication

Figure A.2: Sample structural diagrams of VCL model of *secure simple bank*

and global operations. In Fig. A.1, packages `CommonTypes` and `RolesAndTasksBank` are containers; all others are ensembles.

- *Uses* edges, represented as hollow-headed arrows, indicate that the current package may use elements defined in other packages. In Fig. A.1b, package `Bank` uses package `CommonTypes`, which is given alias `CT`.
- In PDs, enclosure denotes *incorporation*. Incorporation means that the structures of the enclosed packages become part of the current package. In Fig. A.1h, `SecBank` incorporates packages `SecForBank`, `BankWithJI` and `TransactionsSwitch`.
- Incorporated packages are connected with edges to resolve conflicts. There are two kinds of such edges: *overrides* and *merges*. Override edges say that certain sets of the source package override those with the same name of the target package in the current package; only sets with no local properties may be overridden. Merge edges say that certain specified sets with the same name from the linked packages are to be merged. Figure A.1f says that set `User` of package `Authentication` overrides `User` of `AccessControl`.
- The current package may extend incorporated sets. This is specified using an *extends list*. In Fig. A.1g, package `SecForBank` extends incorporated set `User`.

A.3.1 Structural Diagrams

SDs define the structures that, together, make the state space of a package. Figure A.2 gives SDs of packages that are part of the VCL model of secure simple bank. The SDs are as follows:

- A SD defines two kinds of primitive sets: *value* and *class*. Class sets (or, simply, classes) define a set of objects with identity; value sets define a set of values (objects without identity). Classes are represented with a bold-line; they include a label `CLASS` on the top-right part of the contour. Value sets are not bold-lined and they include the label `VALUE`. In SDs of Fig. A.2, `Customer`, `Account`, `User` and `Session` are class sets; all others are value sets.

- Sets whose name are preceded by symbol ◯ are defined in terms of what they have inside them. The ◯ means that the set is *closed* from a definitional point of view because it is defined by what it has inside only. Sets that only include the ◯ are *definitional*. Sets that include symbol ◯ followed by symbol ↔ and that contain the label `DERIVED` are *derived* sets; they are defined from primitive entities of the model. In Figs. A.2b and A.2c, sets `CustType`, `AccType`, `LoginResult` and `UserStatus` are *definitional* value sets describing enumerations, defined by enumerating their possible values. In Fig. A.2a, `TimeNat` is a derived set defined from `Nat`, the set of natural numbers [1].

- A SD can contain constants of *sets* and *scalars*. Constants have their labels preceded by symbol ©. Scalar constants are represented as objects (rectangles); set constants as rounded contours . When placed inside sets, constants objects are used to name values or objects of the enclosing set; this is a common idiom to define enumerations in VCL (e.g. sets `CustType`, `AccType`, `LoginResult` and `UserStatus` of SDs of Fig. A.2 are defined this way). A SD can also include constants with a type designator; such constants are *local* when attached to some set and *global* when they stand alone. In Fig. A.2c, `maxPwMisses` defines a scalar constant of type `Nat` that is local to set `User`.

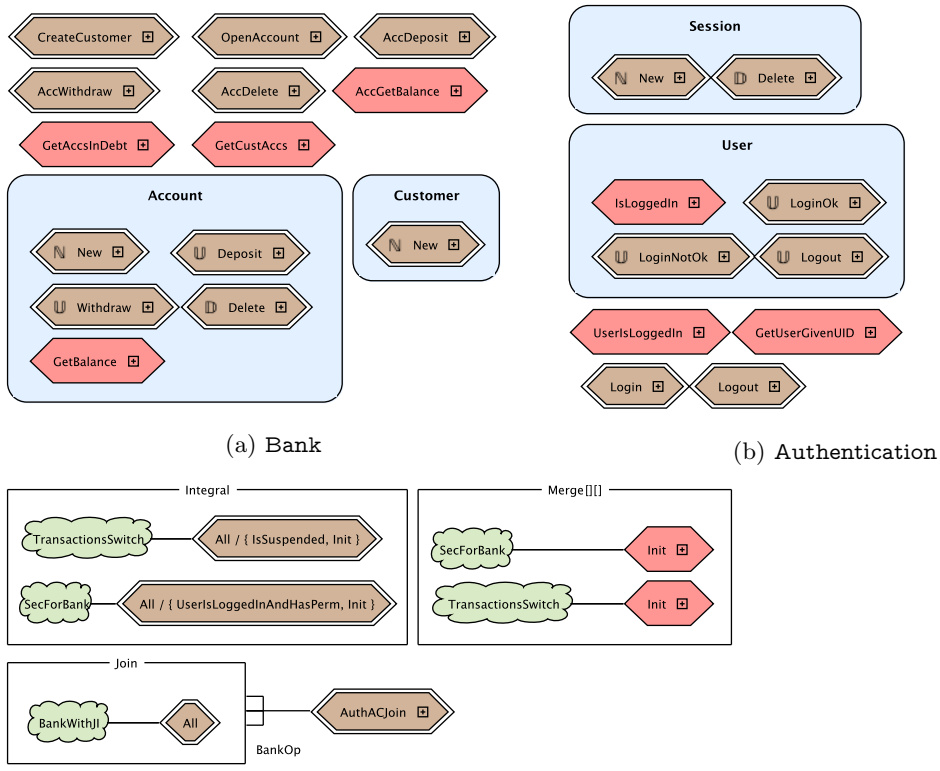
- *Reference* sets contain symbol ↑ and label `REFERENCE`; they are used to refer to sets defined in used packages as described in the PD. Reference sets of Figs. A.2b and A.2c refer to `Name` and `TimeNat` defined in package `CommonTypes` (alias `CT`).

- Edges with circled labels are *relation edges*; they define binary relations between sets and they include multiplicity constraints on each edge's extremity, which are represented both numerically and using lines and circles. A circle on the edge means multiplicity 0, a single line means multiplicity 1, and a line-end with a crow's foot means multiplicity many. In Figs. A.2b and A.2c, `Holds` and `HasSession` are relation-edges defining, respectively, multiplicities one-to-many and one-to-optional (? means optional or the 0..1 UML multiplicity).

- There are two alternative ways of representing state properties (attributes or fields) possessed by all objects of some set: (i) either using directed edges (outgoing arrows) as in SD of Fig. A.2b (e.g. `name`) from the set to which they belong to the set that gives their type, where the edge's label is the property's name, or (ii) listed in the set's properties compartment (according to the syntax, '<name> **:** <type>') in a style that is akin to UML class diagrams, as in Fig. A.2c (e.g. `uid`).

- Assertions, represented as single-lined elongated hexagons, identify invariants, which are separately defined in ADs. Assertions connected to some set are *local*; those standing-alone are *global*. In VCB, double-clicking on an assertion takes the user to its AD (symbol ⊞). In Figs. A.2b and A.2c, `CorporateHaveNoSavings`, `HasCurrentBefSavings` and `HasSessionIffLoggedIn` are global, `SavingsArePositive` and `MaxPwMissesInv` are local.

[1] This defines time as set of time points that are isomorphic to the natural numbers.



(a) Bank

(b) Authentication

(c) SecBank

Figure A.3: Sample behaviour diagrams of VCL model *secure simple bank*

A.4 Behaviour Diagrams

A BD defines a map over the behavioural units of a package; it identifies these units without actually defining them. A BD may include: (a) references to package operations to be separately specified in ADs and CDs or (b) operation compositions. Figure A.3 presents sample BDs of secure simple bank; the operations included in them are as follows:

- There are two kinds of specified operations: *update* (or modifiers) and *observe* (or queries). Queries are represented as assertions, modifiers as contracts (double-lined elongated hexagons). In Fig. A.3b, operations `UserIsLoggedIn`, `GetUserGivenId` and `IsLoggedIn` are queries; all others are modifiers.

- Specified operations may be *local* or *global*: local when they are inside some set and global otherwise. The global operations of a package define the behaviour that the package offers to the outside world. Each specified operation needs to be defined in a AD or CD; double-clicking takes the user to its definition (symbol ⊞). In BDs of Fig. A.3, the operations inside the contours `Account`, `Customer`, `Session` and `User` are local; all others are global.

To support coarse-grained separation of concerns, VCL is equipped with composition mechanisms to put together behaviours from different packages. VCL's composition mechanisms

build-up on ideas surrounding aspect-orientation. BDs support three kinds of operation compositions [AKMG10]: *integral*, *merge* and *join extension*. Such compositions are defined in boxes; each box is named after the kind of composition. The operation compositions of Fig.A.3 are as follows:

- Operations of imported packages are visible within the package, but not visible to the outside world. Integral extension promotes operations from incorporated packages to package operations to make them visible to the outside world. The promoted operation is made available in the new package unaltered (hence name *integral*). Operations to be integrally extended are represented inside an integral extension box (there is at most one); they are represented visually as normal operations connected to the package where they come from. Special operation `All` refers to all operations of a package and it may include a list of operations to exclude. BD of Fig. A.3c says that all operations of package `TransactionsSwitch` with the exception of `IsSuspended` and `Init` are to be integrally extended; likewise for all operations of package `SecForBank` except `UserIsLoggedInAndHasPerm` and `Init`.

- Merge extension is a form of composition that merges or joins separate behaviours coming from different packages. Merge extensions are specified in a merge box; all separate operations that are included in the merge box with the same name are merged into a new package operation joining the separate behaviours. Semantically, merged operations are combined using an operator that is akin to *logical conjunction*. BD of Fig. A.3c includes a merge box saying that `Init` of `SecForBank` is to be merged with `Init` of `TransactionsSwitch`.

- Join extension is VCL's aspect-oriented like mechanism. It inserts a certain extra behaviour into a group of operations. Join extensions involve placing the group of operations to extend inside a join box; the extra behaviours to insert are specified as *join contracts*, comprising a pre- and a post-condition, that is connected to the box through a *fork edge*. BD of Fig. A.3c includes a join extension that says that all operations of `BankWithJI` are to be joined with contract `AuthACJoin`.

A.5 Assertion Diagrams

ADs describe predicates over a single state of the modelled system. They are used to describe invariants and observe operations (queries). Sample ADs are given in Figs. A.4 and A.5; they are as follows:

- An AD is made of two compartments: declarations (top) and predicate (bottom). This is illustrated in ADs of Figs. A.4 and A.5.

- An AD may have a global or a local scope. Global ADs include names of package and assertion; local ADs include names of package, set and assertion. ADs of Figs. A.4c, A.4a and A.5a are local; all others are global.

- The declarations compartment may include variables, which are either scalar (represented as objects) or collections (represented as sets). Each variable has a name and a type. Variables can denote either inputs (name suffixed by '?') or outputs (name suffixed by '!'). Figure A.5d declares output set `accs!`; Figs. A.5a, A.5b and A.5c declare several input and output objects.

- The declarations compartment may include imported assertions, either standing alone or combined in logical formulas. Double-clicking on an imported assertion takes the user to

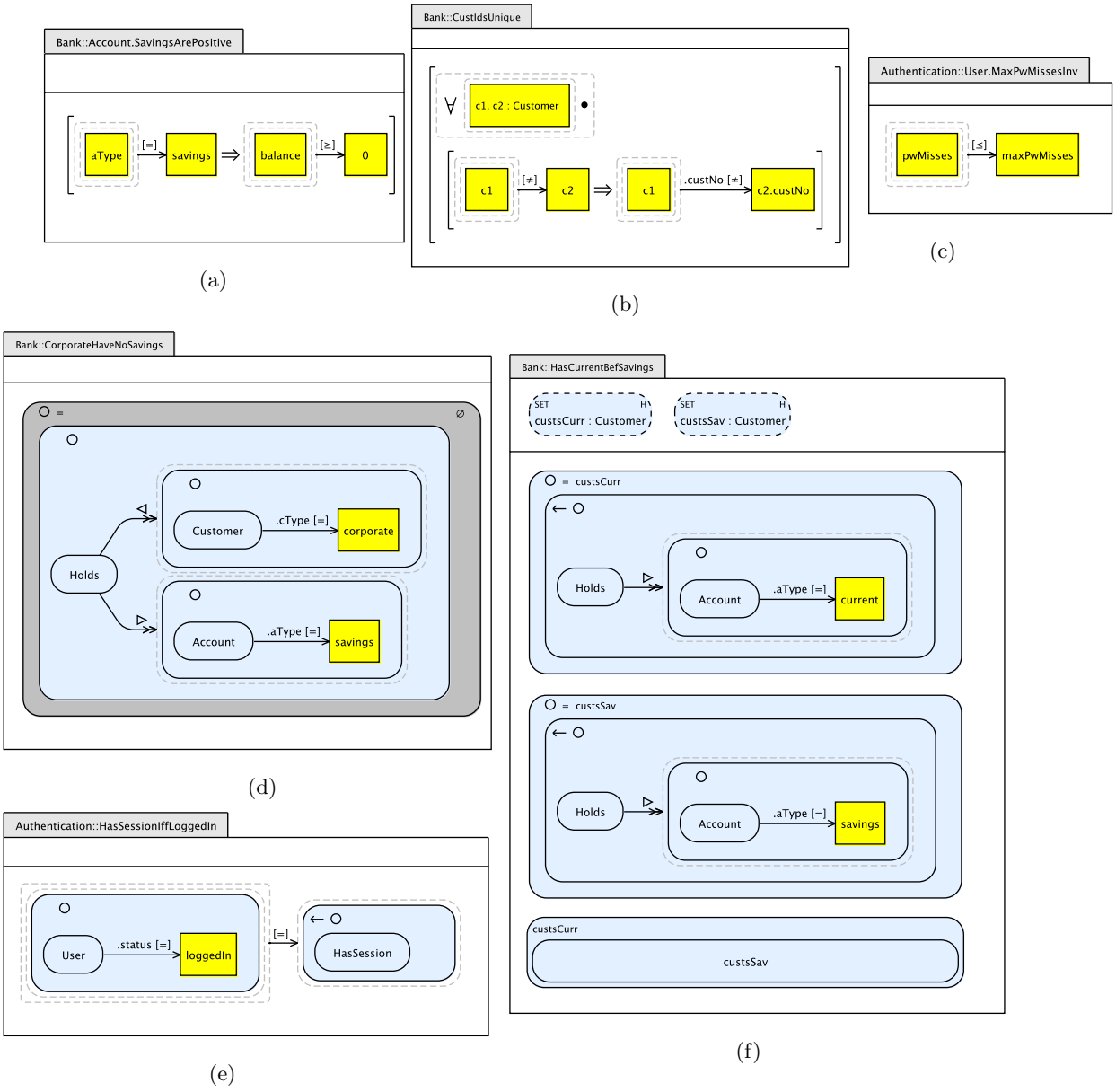


Figure A.4: Sample ADs of invariants of VCL packages `Bank` and `Authentication` of secure simple bank

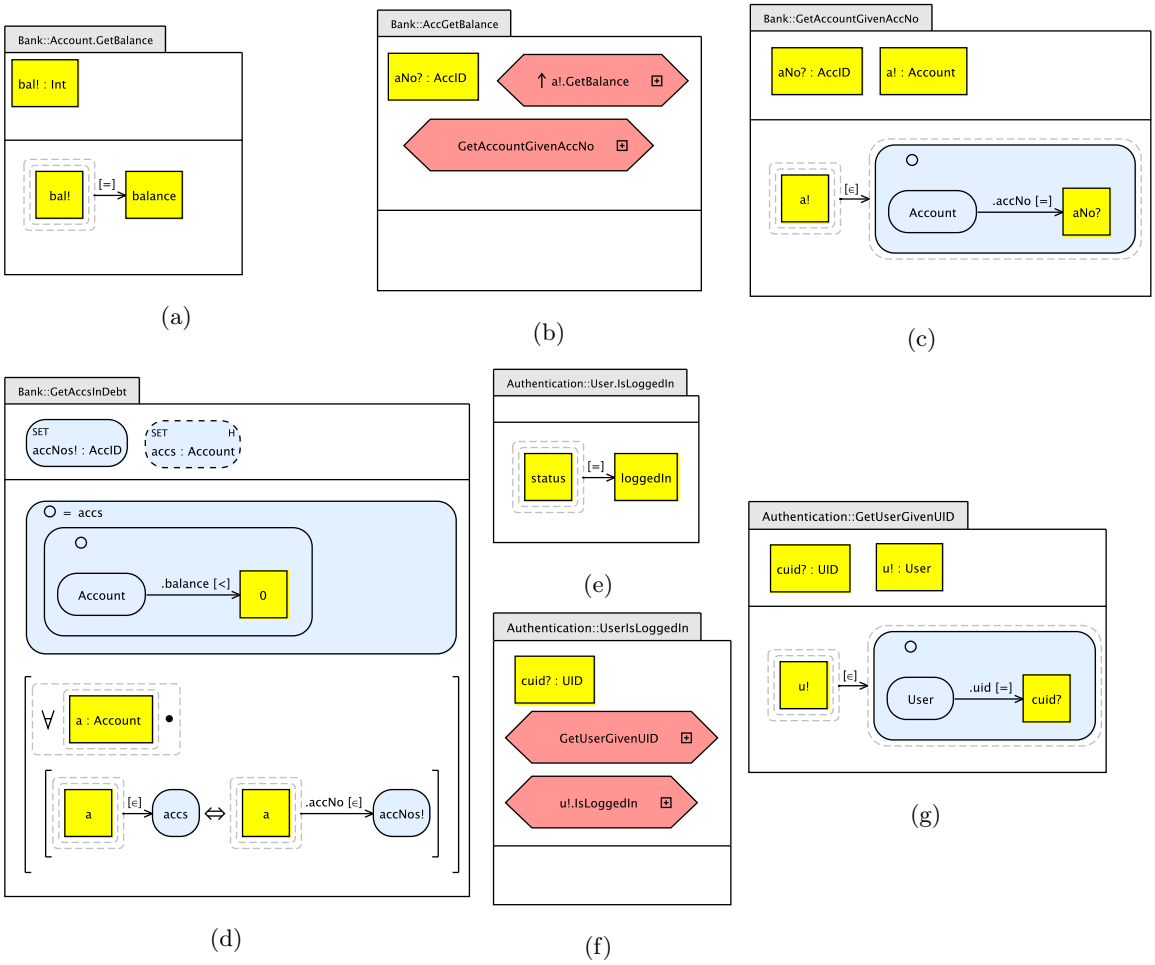


Figure A.5: Sample ADs of operations of VCL packages `Bank` and `Authentication` of secure simple bank

its AD definition (symbol ⊞). An assertion import comprises an optional up arrow symbol ↑, name of imported assertion with optional origin qualifier and an optional rename list. ↑ symbol indicates that the import is total (variables and predicate are imported); when not present the import is partial (only the predicate is imported). Rename list indicates variables of imported assertion that are to be renamed (e.g. $[a!/a?]$ says that $a!$ replaces $a?$). Fig. A.5b has two assertion imports: one total and one partial. Total assertion import says that assertion `GetBalance` is to be called on `Account` object `a!`; as import is total, variable *bal*! defined in `Account.GetBalance` (Fig. A.5a) is also defined in `AccGetBalance`. Partial import of `GetAccountGivenAccNo` means that output $a!$ is visible in `AccGetBalance`, but not to the outside world; input `aNo?` from `GetAccountGivenAccNo` is visible to the outside world as it is also declared in `AccGetBalance`.

- The predicate compartment includes visual formulas made up of set expressions, predicates and propositional logic operators. Figure A.5a expresses an equality predicate using a predicate property edge to say that `bal!` is to hold value `Account.balance`. Figure A.4a

expresses an implication formula to say that if the property `aType` of `Account` has value `savings` then the property `balance` must be greater or equal than 0. Figure A.5d outputs the set of account numbers whose balances are negative; this builds a set using a set definition construction (symbol ◯) by constraining set `Account` using predicate property edges (arrows); the constructed set (`accs`) is then used to say, using a quantifier, that the output set `accNos!` contains all account numbers of the accounts in `accs`. Figure A.5c expresses a set membership predicate using a predicate property edge to say that output `a!` belongs to accounts with property `accNo` equal to `aNo?` (there is at least one). Figure A.4d comprises a set formula that defines a set by constraining the relation `Holds`, using property edge modifiers with operators domain (◁) and range restriction (▷), to give the set of tuples made of corporate customers and savings accounts; outer shading (reinforced with symbol ∅) then says that this set must be empty.

A.6 Contract Diagrams

CDs describe system dynamics. They comprise a pair of predicates corresponding to pre- and post- conditions. The pre-condition describes what holds before the operation is executed. The post-condition describes the effect of the operation, saying what holds after execution. CDs are used to describe operations that change the state of the modelled system. Sample CDs are given in Fig. A.6; they are as follows:

- Like ADs, CDs are made of two principal compartments for declarations and predicate. In CDs, the predicate compartment is subdivided in two for pre-condition (left) and post-condition (right). This is illustrated in CDs of Fig. A.6.
- CDs are similar to ADs in terms of what can be included in the declarations compartment. The only difference is that CDs can include both imported assertions and contracts. This is illustrated in CDs of Fig. A.6. In Fig. A.6e, import of contract `HasSessionAddNew` includes a renaming. In Fig. A.6f, the two imported contracts are combined using a disjunction to say that a login operation is either successful (`LoginOk`, Fig. A.6e) or not (`LoginNotOk`, Fig. A.6d).
- In CDs, pre- and post- condition compartments are made of the same sort of visual formulas used in the predicate compartment of ADs. In the post-condition compartment, the variables that refer to the after state are bold-lined. In Figs. A.6a and A.6c, pre- and post-conditions compartments are made of arrows formulas stating equality predicates; the post-state variable `status` in Fig. A.6c is bold-lined in the post-condition compartment to say that its value changes from `loggedIn` (precondition) to `loggedOut` (post-condition).

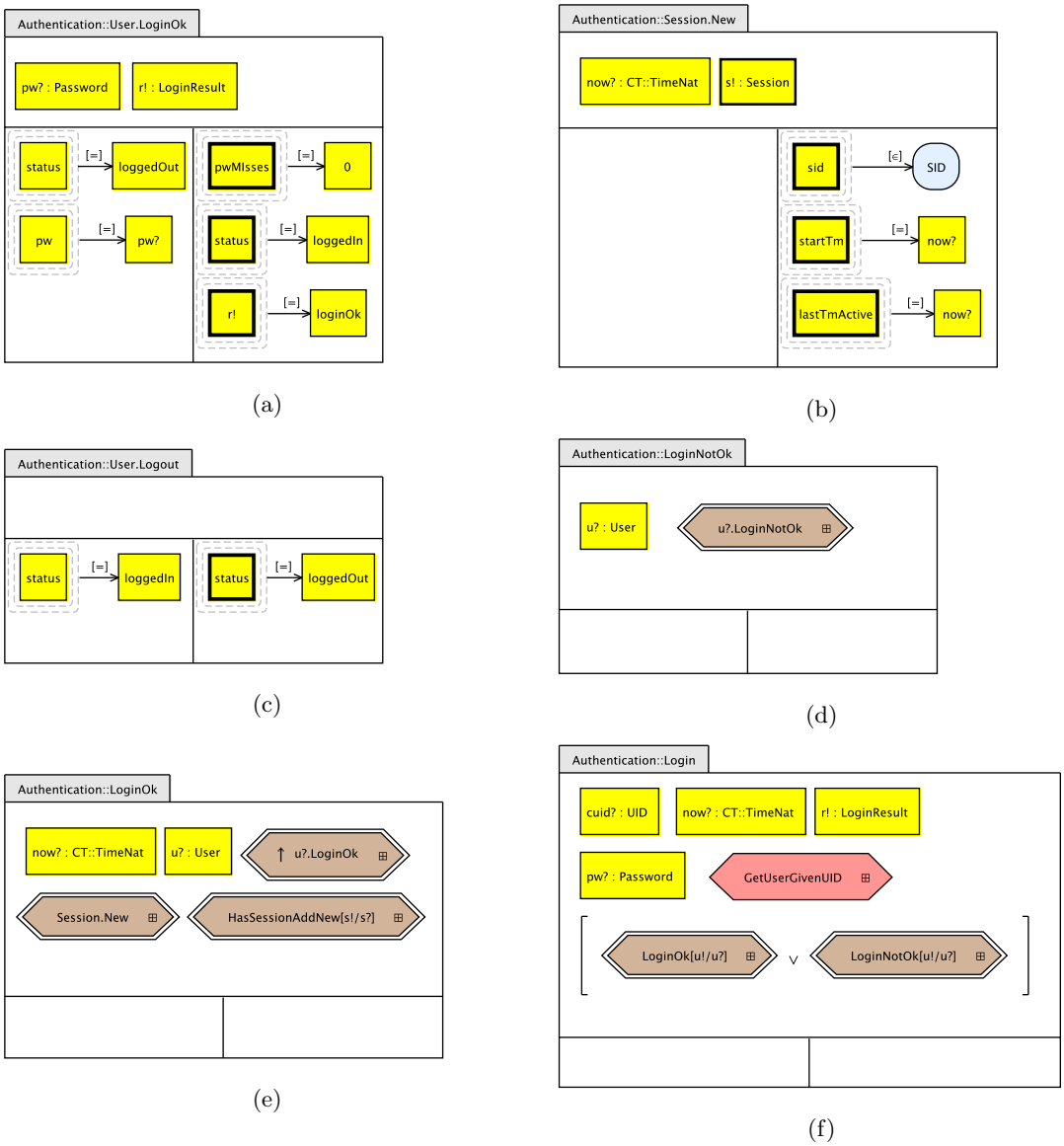


Figure A.6: Sample CDs of package `Authentication` of secure simple bank

Appendix B

Experiment Case Studies

This appendix presents the description of the case studies used in the experiment presented here. These descriptions were distributed to subjects and the tasks of the experiment were based on them.

B.1 University Library

A university library needs to manage books, library users and the books that members borrow. The aim is to provide a system to manage the library's catalogue, to enable members to search for books and find availability information on the book's copies, and to give members detailed information on their borrowings.

The system needs to record information that is associated with books, namely: a book number, a title, an ISBN, a year of publication and several authors; an author has a name; no two books may have the same number. A book may have several copies in the library; members borrow copies of books from the library; a copy is associated with one book only.

A member has a name and a library number; library numbers are distinct. The library considers three categories of members: *taught students*, *research students* and *staff*. Each category has different borrowing entitlements: taught students can borrow at most 10 books at any one time, research students 15, staff 25.

Each book copy has its own identifier provided by the library (the copy number); no two copies may have the same number. There are three kinds of copies, which set different borrowing restrictions: *one-week*, *one-month* and *restricted*. One-week and one-month copies can be borrowed for one-week and one-month periods, respectively; restricted copies can be consulted in the library's premises, but they cannot be borrowed. Members may borrow non-restricted copies of books whose status is *shelved*; once a member borrows a copy, the system places it in the status *on-loan* and the due date for the copy is updated according to its loan period (one-week or one-month). The status of restricted copies is always *shelved*.

The status of a copy is changed back to *shelved* when the member returns the copy and there is not an active recall set on the copy. Members may renew their borrowings up to a maximum of 3 renewals per borrowing, provided their copy has not been recalled and the copy is not overdue. Once the user renews a borrowing the due date is updated according to the copy's loan period.

A member may recall a book copy that is being borrowed, provided there are no other copies of the recalled book available for borrowing and that the member is not the copy's borrower. When a copy is recalled its status becomes *recalled* and the return due date is reduced to at

most one week (one-week if the current due-date is after one week from the current date or the current due date otherwise). Once a recalled copy is returned, its status becomes *toCollect*.

In this first phase, the system shall provide the following operations:

- *Borrow.* Given a member's library number and a copy number, the system records the borrowing.
- *Return.* Given the number of a copy, the system records the return of the copy to the library.
- *Renew.* Given the number of a copy, the system renews its borrowing.
- *Recall.* Given a member's library number and the number of the copy to recall, the system recalls that particular copy that is being borrowed for that particular member.
- *Collect.* The user collects a previously recalled copy that became available for collection in the library.
- *GetBorrowedCopies.* Given a member's library number, the system yields the copies being borrowed by that particular member.
- *GetActiveRecalls.* Given a member's library number, the system outputs the copies that the member has recalled, but have not been returned to the library.
- *GetWaitingCollection.* Given a member's library number, the system outputs the copies that are awaiting collection for the member following a recall.

B.2 Flight Booking

The company *EasyPeasyFlightBooking.com* intends to put in place an online and automated flight reservation system for scheduled flights. A scheduled flight has a flight specification, which comprises an airline carrier, a flight number, one originating and one destination airport, and one departure and one arrival time; originating and destination airports of a scheduled flight must be different; no two scheduled flights may have the same flight number. An airline has a name and a code (e.g 'LG', 'TAP', 'BA', 'AF'); no two airlines can have the same code. An airport has a name and a code (e.g. 'Lux' for Luxembourg's city airport, 'LGW' for London Gatwick); no two airports can have the same airport code.

An actual flight is associated with a flight specification. It has a date and provides reservable seating for passengers. There cannot be flights with the same number on the same date.

Passengers may reserve flight seats using the system. Each reservation comprises one flight seat only. A seat may have one of the following classes: *economy*, *economyPlus*, *first* or *business*. Each seat has a number of available fare classes, namely: *superSaver*, *saver*, *flex*, *relax* and *relaxPlus*. The *superSaver* and *saver* fares applies to economy seats only; the fare *relaxPlus* applies to seats of class *business* and *first* only. A seat may have two possible status: *free* or *booked*. Only seats whose status is *free* may be reserved. When the user decides to reserve a free seat, its status becomes *booked*.

When a seat is reserved, the system records the reservation of the seat for the passenger. A reservation includes the reservation number, the date of the reservation, passenger's name, the passenger's chosen fare class, the agreed fare, the currency of the agreed fare (fares include *EUR*, *USD* and *GBP*), the penalty for changing or cancelling a flight. No two reservations can have the same number. A reservation can have the following status: *pending*, *hold*, *confirmed* or *void*.

When the passenger reserves the seat, the reservation becomes *pending*. Then, when the airline confirms the reservation the status changes to *hold*. It is only when the passenger pays for the reservation that the status becomes *confirmed*. A reservation acquires the status *void* when it is cancelled; as a result, the seat associated with the reservation becomes *free*.

In the first phase, the system shall have the following operations:

- *ReserveFlightSeat.* This operation receives a flight number, a date, the passenger name, a chosen fare class and the chosen seat class, the system reserves the selected flight seat for the passenger. The chosen date must be in the future.
- *AirlineConfirmsBooking.* Given a reservation number, the system records the fact that the airline confirmed the passenger's booking.
- *AirlineConfirmsPayment.* Given a reservation number, the system records the fact that the passenger's payment has been confirmed by the airline.
- *CancelReservation.* Given a reservation number, the system cancels the reservation for the flight.
- *ChangeDateOfReservation.* Given a reservation number and a date, the system changes the reservation to another seat for the same flight number on the given date. This may happen provided the current date is before the date of the current reservation, the reservation's fare class is *flex*, *relax* or *relaxPlus*, the status of the reservation is *confirmed* and there is an available seat of the reserved class for the selected flight.
- *ChangeFlightAndDateOfReservation.* Given a reservation number, a date, and flight number, the system changes the reservation to another flight possibly of another carrier. This may happen provided the reservation's fare class is *relax* or *relaxPlus*, the current date is before the date of the reserved flight, the status of the current reservation is *confirmed*, and the new seat is of class *economy*.
- *GetFreeSeatsForFlightOfClass.* Given a flight number, a date and a seat class, the system retrieves all free seats for the chosen flight with the given class.

Appendix C

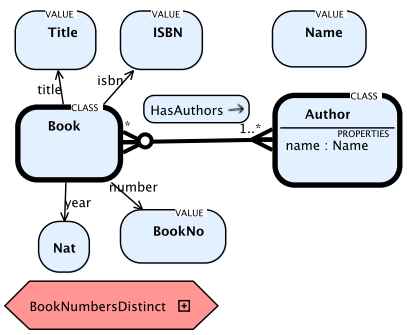
University Library Materials

This chapter presents all materials that support the tasks of the experiment and that are related to the university library case study.

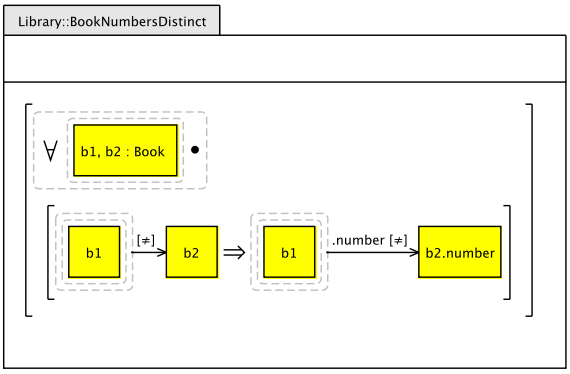
C.1 Starting Models

For the task on model construction of the state space, subjects were given partial models with some structures and invariants, and they were asked to complete the definition of state space (structures and invariants). The supplied models are as follows:

- The starting VCL model comprises an incomplete SD and the AD of one invariant (Fig. C.1).
- The starting UML+OCL model comprises an incomplete class diagram (Fig. C.2).



(a) Structural diagram



(b) AD of invariant `BookNumbersDistinct`

Figure C.1: The starting VCL model of the university library case study

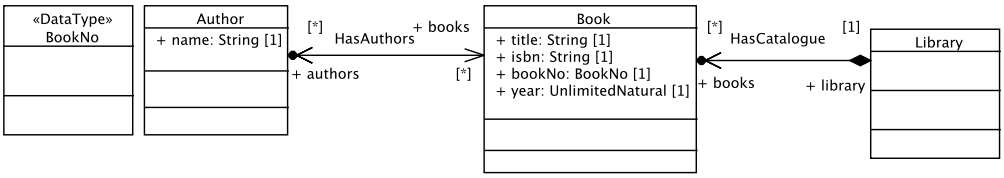


Figure C.2: Class diagram of the university library case study

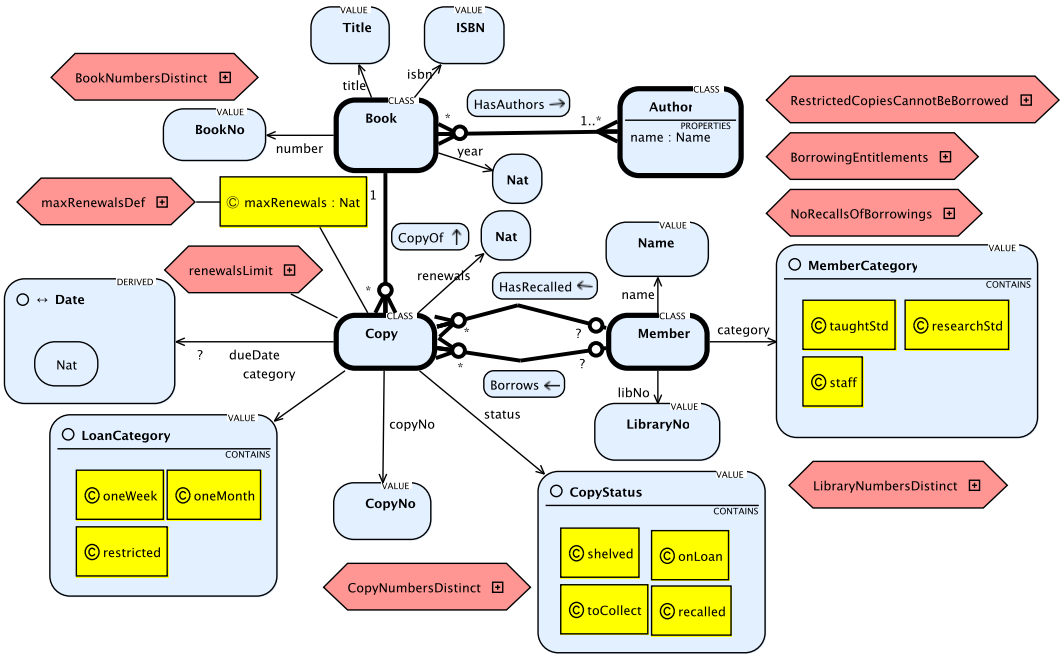
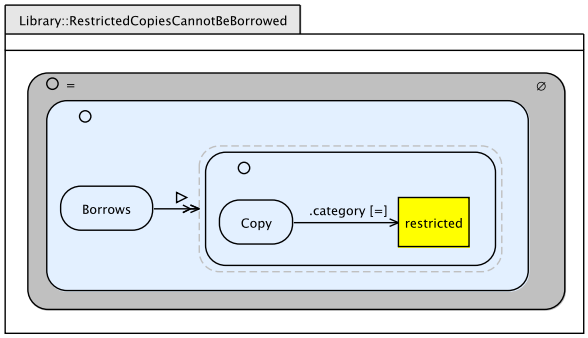


Figure C.3: Structural diagram of the intermediate model of the University library system

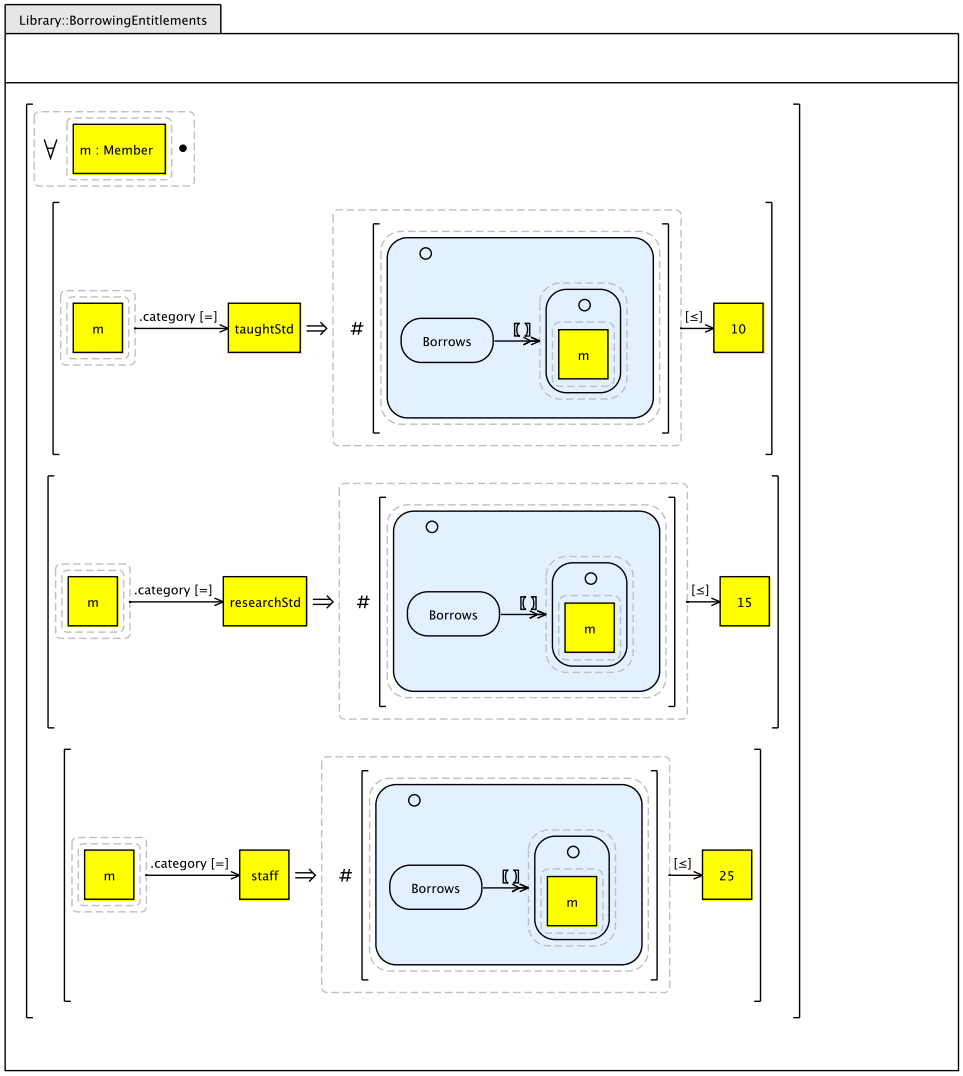
C.2 Intermediate Models

For the task on model construction of operations, subjects were given a complete state space model and a partial behavioural model and they were asked to model some operations of the system. The supplied models are as follows:

- The VCL state space model comprises the SD of Fig. C.3 and the ADs of Fig. C.1b together with ADs of Figs. C.4 to to C.5. The partial VCL behavioural model comprises the BD and AD of Fig. C.6.
- The UML+OCL model comprises the class diagram of Fig. C.8 and the OCL invariants and operations of Fig. C.9.

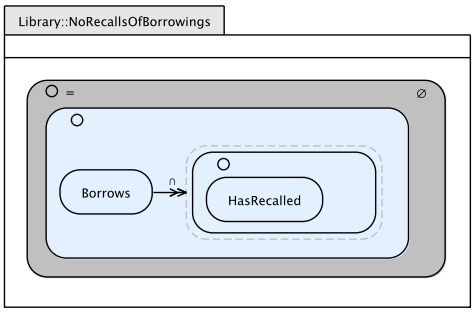


(a) AD of invariant RestrictedCopiesCannotBeBorrowed

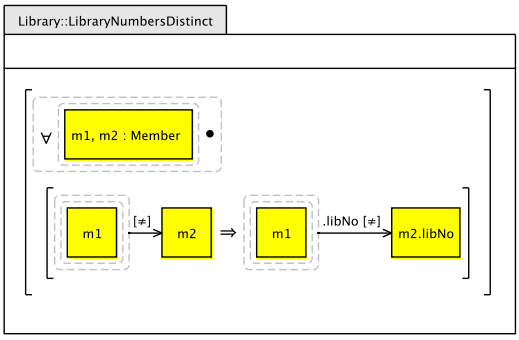


(b) AD of invariant BorrowingEntitlements

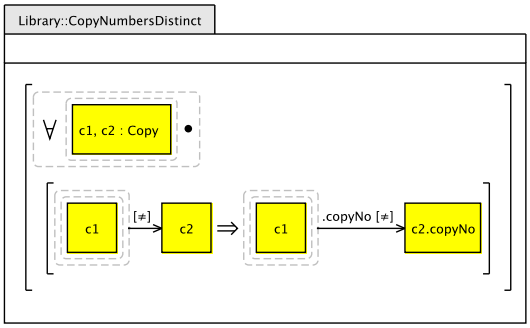
Figure C.4: ADs of invariant of the University library



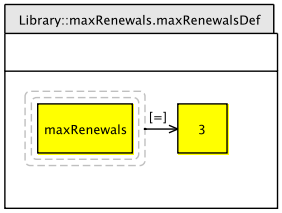
(a) AD of invariant NoRecallsOfBorrowings



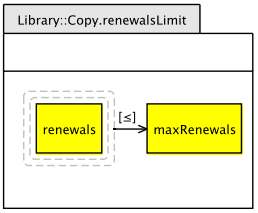
(b) AD of invariant LibraryNumbersDistinct



(c) AD of Invariant CopyNumbersDistinc

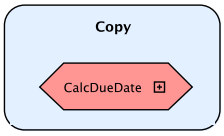


(d) AD of Invariant Copy.maxRenewals

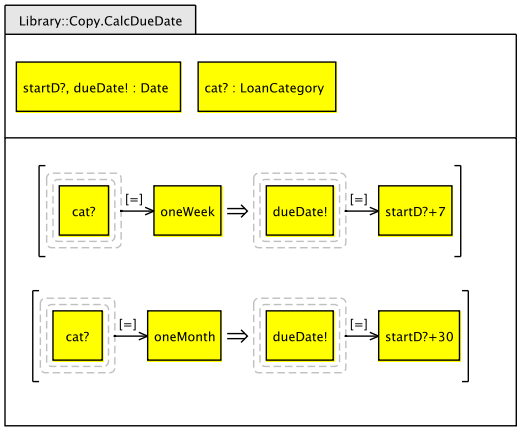


(e) AD of Invariant Copy.renewalsLimit

Figure C.5: ADs of invariants of university library



(a) BD



(b) AD of Operation Copy.CalcDueDate

Figure C.6: Intermediate VCL Behavioural model of university library

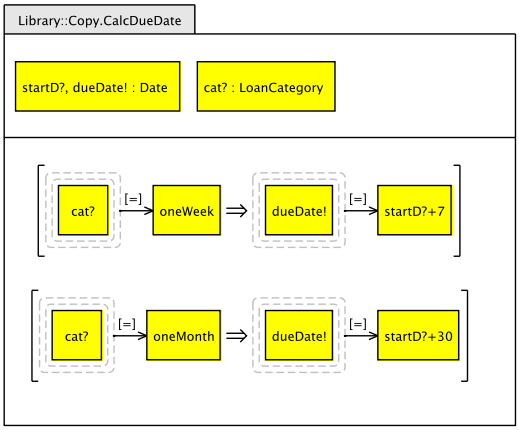


Figure C.7: AD of Operation Copy.CalcDueDate

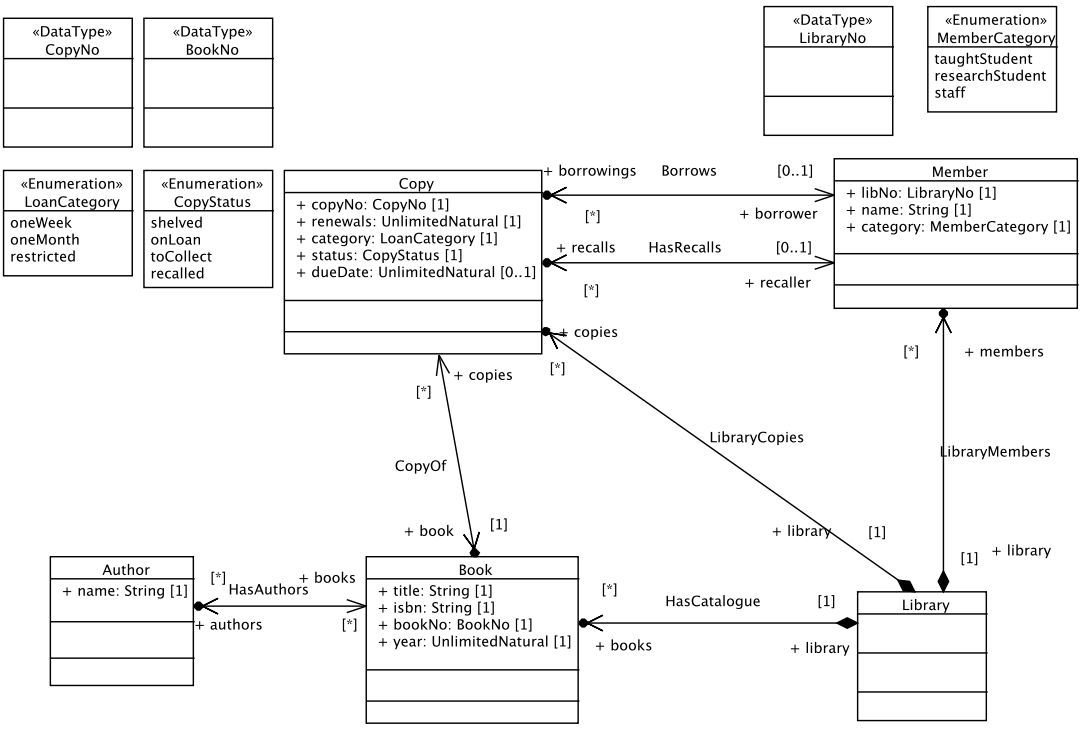


Figure C.8: Class diagram of the university library case study in the intermediate model

```
context Member
inv: category = MemberCategory::taughtStudent implies borrowings->size() <= 10
  and category = MemberCategory::researchStudent implies borrowings->size() <= 15
  and category = MemberCategory::staff implies borrowings->size() <= 25
```

(a) OCL invariant *Member* :: *BorrowingEntitlements*

```
context Member
inv: borrowings->intersection(recalls)->isEmpty
```

(b) OCL invariant *Member* :: *NoRecallsOfBorrowings*

```
context Copy
inv: self.renewals <= 3
```

(c) OCL invariant *Copy* :: *MaxRenewals*

```
context Copy
inv: category=LoanCategory::restricted
  implies status = CopyStatus::shelved
```

(d) OCL invariant *Copy* :: *RestrictedNotBorrowable*

```
context Copy::calcDueDate(now : UnlimitedNatural) : UnlimitedNatural
pre: category <> LoanCategory::restricted
post: category@pre() = LoanCategory::oneWeek implies result = now + 7
  and category@pre() = LoanCategory::oneMonth implies result = now + 30
```

(e) OCL operation *Copy* :: *calcDueDate*()

Figure C.9: OCL invariants and operation of university library

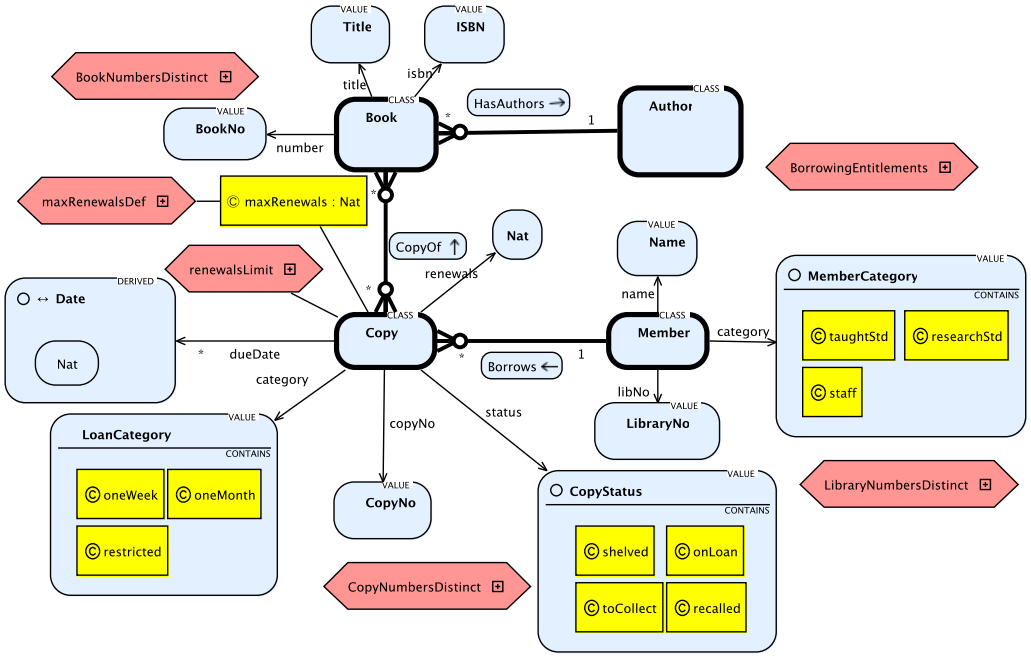
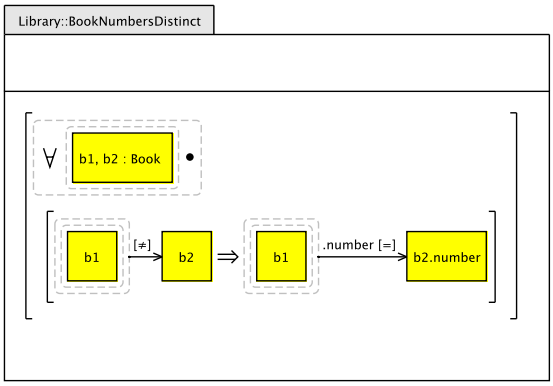


Figure C.10: Structural diagram of VCL model of university library with seeded errors

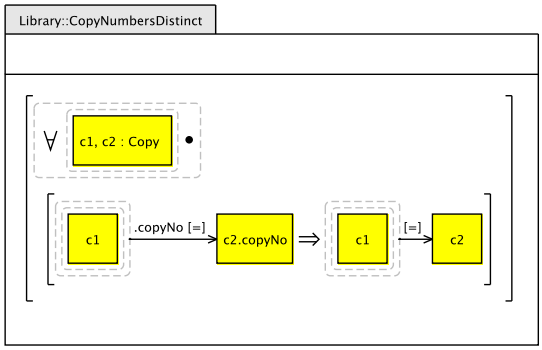
C.3 Defect detection Models

The models with seeded errors for the defect detection task are as follows:

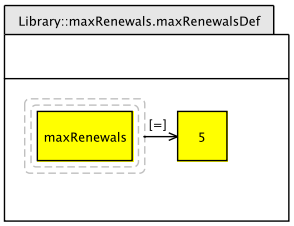
- The state space VCL model is given in Figs. C.10 to Fig. C.12. The behavioural VCL model is given in Figs. C.13 and C.14.
- The structural UML+OCL model is given in Figs. C.10 to C.16, and the behavioural model is given in Figs. C.18 to C.26.



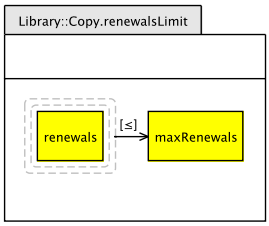
(a) Invariant BookNumbersDistinct



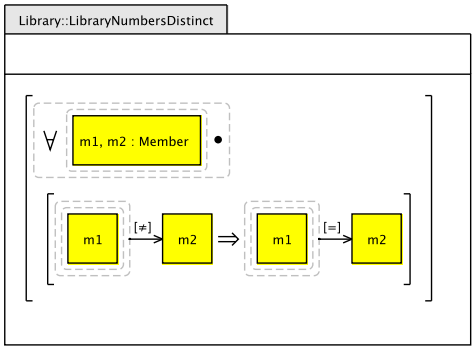
(b) AD of invariant CopyNumbersDistinct



(c) Invariant Copy.maxRenewals



(d) Invariant Copy.renewalsLimit



(e) Invariant LibraryNumbersDistinct

Figure C.11: ADs of invariants of the VCL model of university library with seeded errors

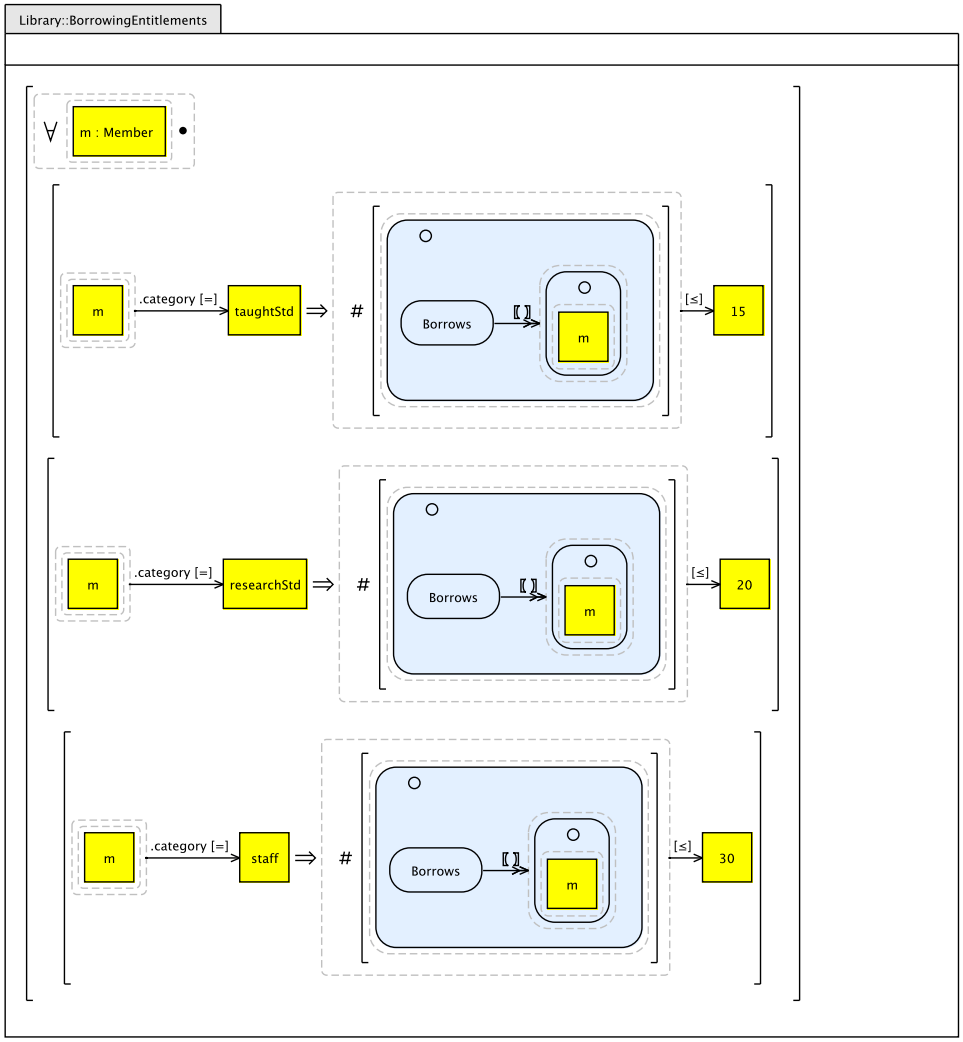
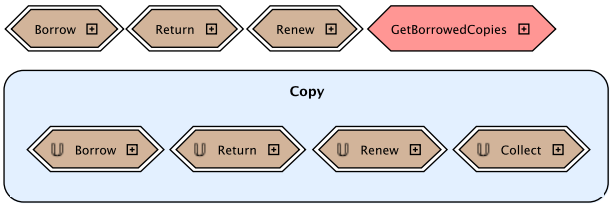
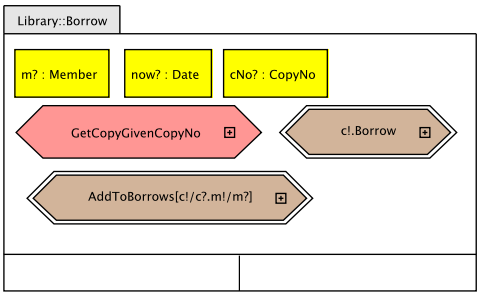


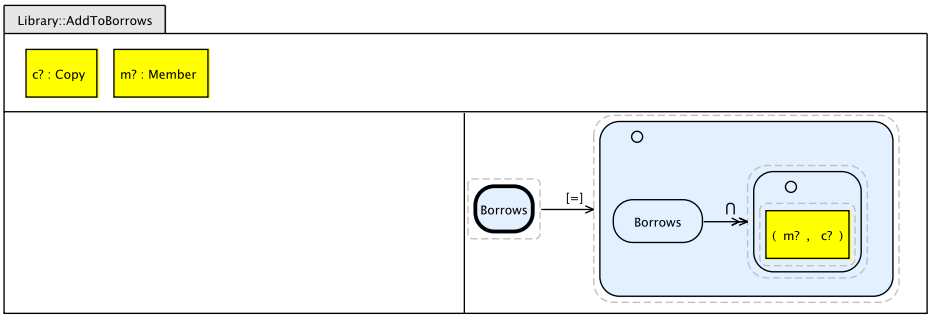
Figure C.12: AD `BorrowingEntitlements` of the VCL model with seeded errors



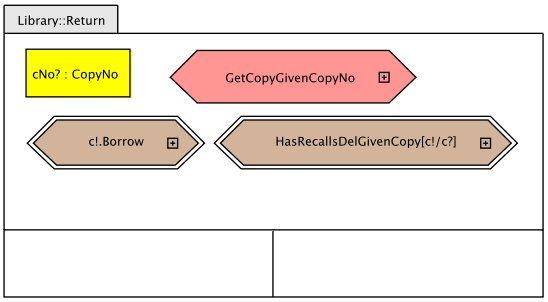
(a) Behavioural diagram



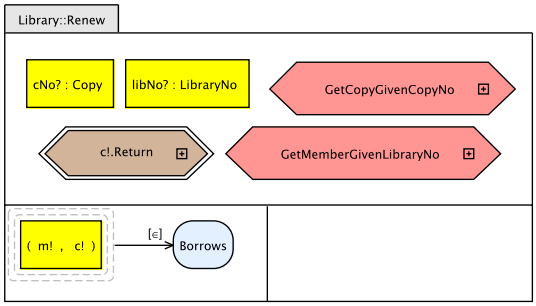
(b) Contract of operation Borrow



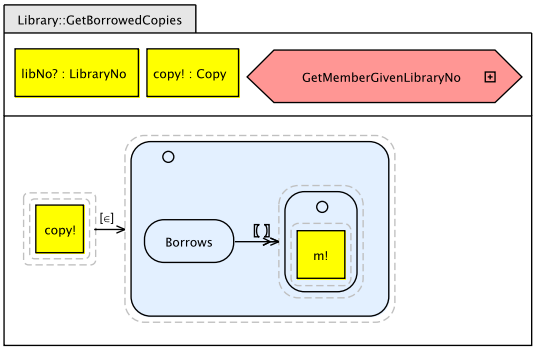
(c) AD of Copy.Return



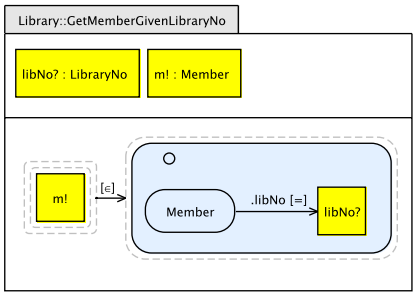
(d) Contract of operation Return



(e) Contract of operation Renew

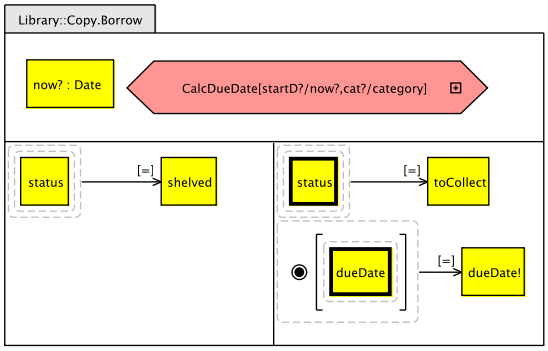


(f) AD of operation GetBorrowedCopies

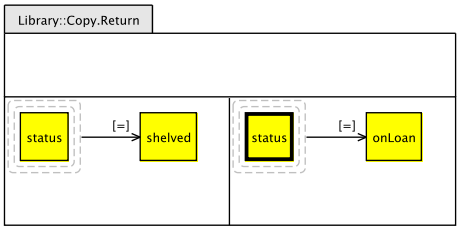


(g) AD of operation GetMemberGivenLibraryNo

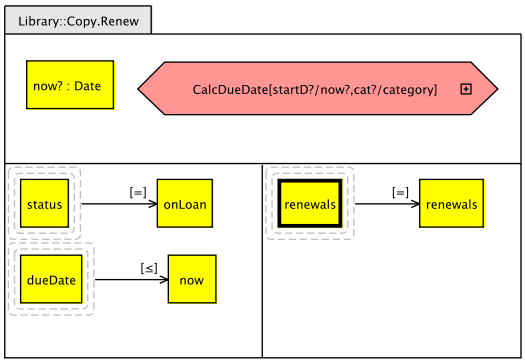
Figure C.13: Global Behavioural VCL model with seeded errors



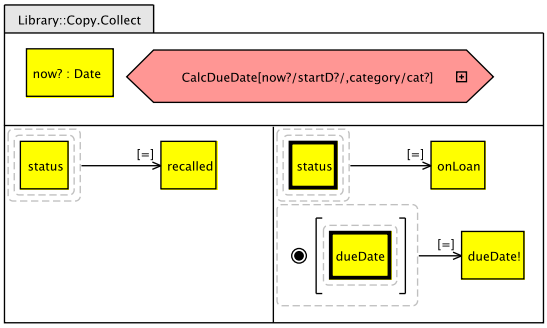
(a) CD of Copy.Borrow



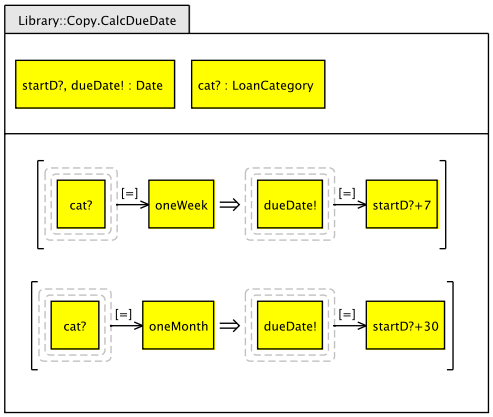
(b) CD of Copy.Return



(c) CD of Copy.Renew



(d) CD of Copy.Collect



(e) CD of Copy.Collect

Figure C.14: Local behaviour of set **Copy** in the VCL model of university library with seeded errors

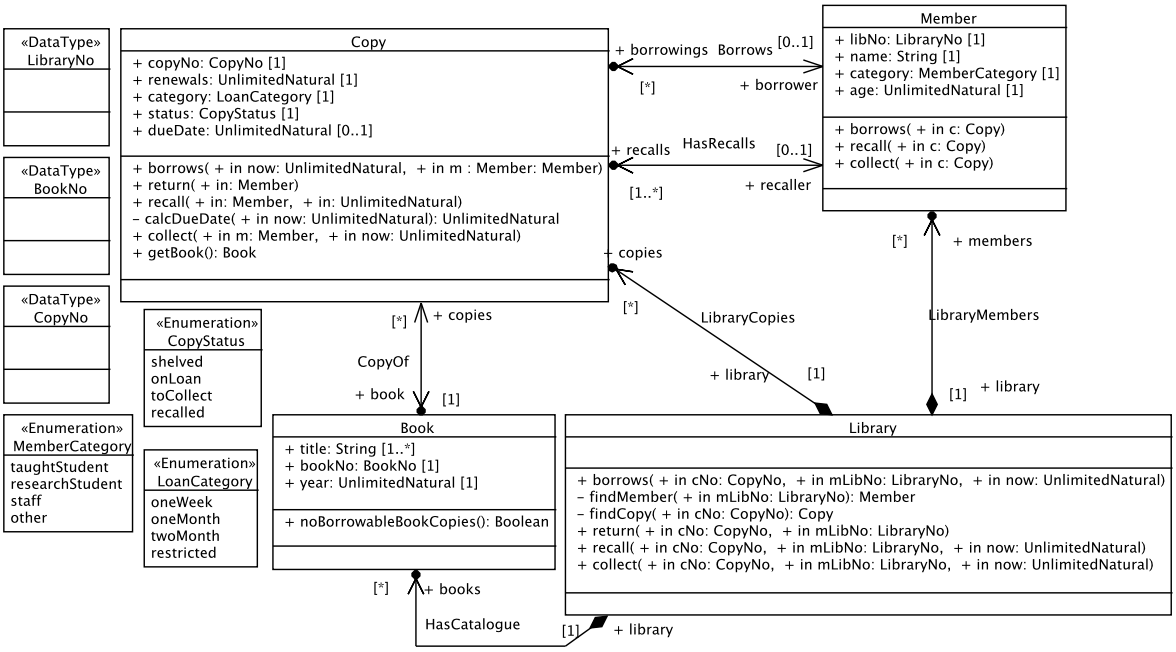


Figure C.15: UML+OCL Class diagram of university library in model with with seeded errors

```
context Member
inv: category = MemberCategory::taughtStudent implies borrowings->size() <= 10
  and category = MemberCategory::researchStudent implies borrowings->size() <= 15
  and category = MemberCategory::staff implies borrowings->size() <= 25
```

(a) OCL constraint *Member* :: *BorrowingEntitlements*

```
context Copy
inv: self.renewals < 3
```

(b) OCL constraint *Copy* :: *MaxRenewals*

```
context Copy
inv: category=LoanCategory::restricted
  implies status = CopyStatus::onLoan
```

(c) OCL constraint *Copy* :: *RestrictedNotBorrowable*

Figure C.16: OCL invariants of the university library case study in the model with seeded errors

```
context Library :: findMember(mLibNo : Library) : Member
post: result = members@pre->any(libNo <> mLibNo)
```

(a) OCL operation *Library* :: *findMember*()

```
context Library :: findCopy(cNo : CopyNo) : Copy
post: result = copies@pre->any(copyNo=cNo)
```

(b) OCL operation *Library* :: *findCopy*()

```
context Library::borrows(cNo : CopyNo, mLibNo : LibraryNo, now : UnlimitedNatural)
post:
  let c : Copy = findCopy(cNo),
      m : Member = findMember (mLibNo)
  in
    c.borrows(now, m) and m.borrows(c)
```

(c) OCL operation *Library* :: *Borrow*()

```
context Library::return(cNo : CopyNo, mLibNo : LibraryNo, now : UnlimitedNatural)
post:
  let c : Copy = findCopy(cNo),
      m : Member = findMember (mLibNo)
  in
    c.return(m) and m.return(c)
```

(d) OCL operation *Library* :: *Return*()

```
context Library::recall(cNo : CopyNo, mLibNo : LibraryNo, now : UnlimitedNatural)
post:
  let c : Copy = findCopy(cNo),
      m : Member = findMember (mLibNo)
  in
    c.getBook()->noBorrowableBookCopies() and c.recall(now) and m.collect(c)
```

(e) OCL operation *Library* :: *Recall*()

```
context Library::collect(cNo : CopyNo, mLibNo : LibraryNo, now : UnlimitedNatural)
let c : Copy = findCopy(cNo),
     m : Member = findMember (mLibNo)
in
   c.collect(m, now)
```

(f) OCL operation *Library* :: *Collect*()

Figure C.17: OCL operations of class `Library` in the model with seeded errors of the university library case study

```
context Member::borrows(c : Copy)
post: borrowings = borrowings@pre()->union(Set{c})
```

(a) OCL operation *Member* :: *borrows*()

```
context Member::recall(c : Copy)
post: recalls = recalls@pre()->union(Set{c})
```

(b) OCL operation *Member* :: *becall*()

```
context Member::collect(c : Copy)
pre: recalls->includes(m)
post: recalls = recalls@pre() - Set{c}
  and borrowings = borrowings@pre()->union(Set{c})
```

(c) OCL operation *Member* :: *collect*()

```
context Book::noBorrowableBookCopies() : Boolean
post: result = copies->forAll(status<>CopyStatus::onLoan or category = LoanCategory::restricted)
```

(d) OCL operation *Book* :: *noBorrowableBookCopies*()

Figure C.18: OCL operations of classes `Member` and `Book` in the model with seeded errors of the university library case study

```
context Copy::borrows(now : UnlimitedNatural, m : Member)
pre: status = CopyStatus::shelved or status = CopyStatus::toCollect
post: status = CopyStatus::onLoan and borrower = m and dueDate = calcDueDate(now)
```

(a) OCL operation *Copy* :: *Borrow*()

```
context Copy::return(m : Member)
pre: borrower = m and dueDate>= now
  and (status=CopyStatus::onLoan or status=CopyStatus::recalled)
post: renewals = 0 and dueDate->isEmpty() and borrower->isEmpty()
  and status@pre() = onLoan implies status = CopyStatus::shelved
  and status@pre() = recalled implies status = CopyStatus::toCollect
```

(b) OCL operation *Copy* :: *Return*()

```
context Copy::recall(m : Member, now : UnlimitedNatural)
pre: status = CopyStatus::onLoan
post: status = CopyStatus::onLoan and calcDueDate(now) < dueDate@pre()
  implies dueDate = calcDueDate(now) and recaller = m
```

(c) OCL operation *Copy* :: *Recall*()

```
context Copy::collect(m : Member, now : UnlimitedNatural)
pre: status = CopyStatus::toCollect and recaller = m
post: status = CopyStatus::recalled and recaller->isEmpty()
  and borrower = m and dueDate = calcDueDate (now)
```

(d) OCL operation *Copy* :: *Collect*()

```
context Copy::calcDueDate(now : UnlimitedNatural) : UnlimitedNatural
pre: category <> LoanCategory::restricted
post: category@pre() = LoanCategory::oneWeek implies result = now + 30
  and category@pre() = LoanCategory::oneMonth implies result = now + 7
```

(e) OCL operation *Copy* :: *calcDueDate*()

```
context Copy::getBook() : Book
post: result = book
```

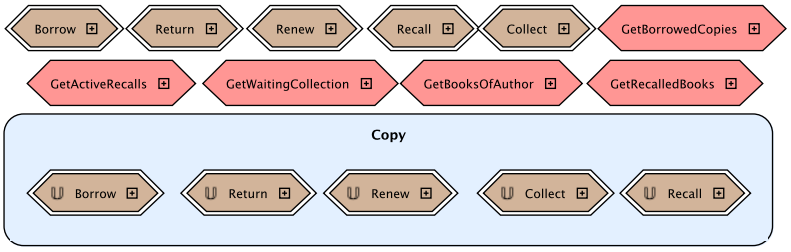
(f) OCL operation *Copy* :: *getBook*()

Figure C.19: OCL operations of class Copy in the model with seeded errors of the university library case study

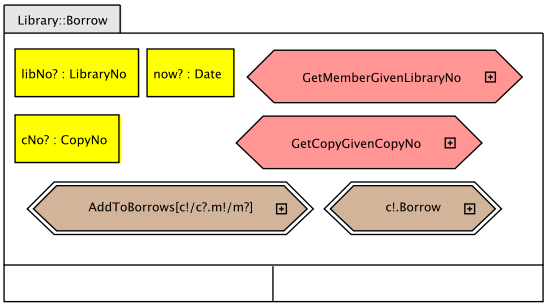
C.4 Solution Models

The solution models were given to subjects as part of the model comprehension task. They are as follows:

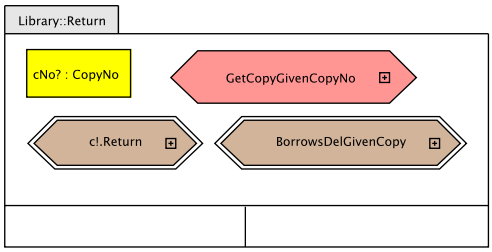
- The VCL solution model consists of the state space model presented in Figs. C.3 to C.4b and the behavioural model is presented in Fig. C.20 to C.24.
- The UML+OCL class diagram is given in Fig. C.25, the invariants are given in Fig. C.9, and the operations (behaviour) are given in Figs. C.26 to C.28.



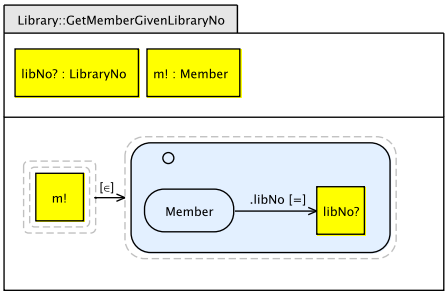
(a) Behavioural diagram



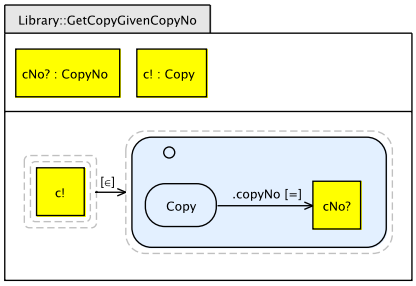
(b) Contract of operation `Borrow`



(c) Contract of operation `Return`

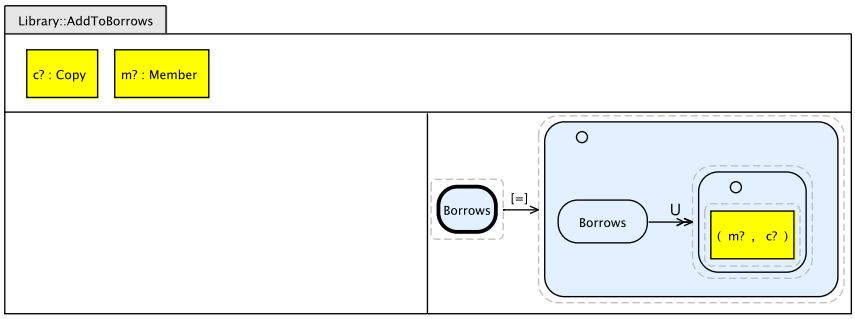


(d) AD of operation `GetMemberGivenLibraryNo`

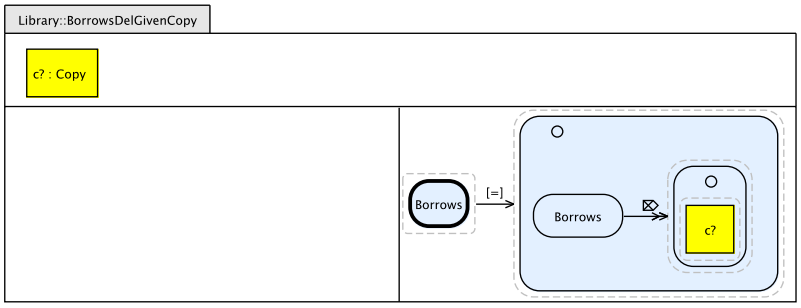


(e) AD of operation `GetCopyGivenCopyNo`

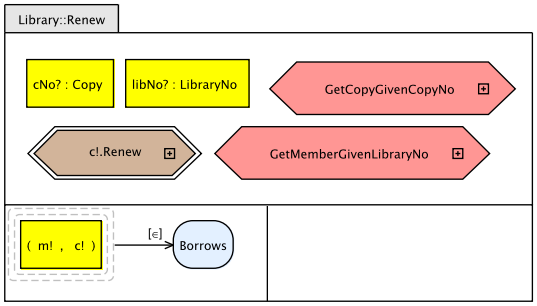
Figure C.20: Global Behavioural VCL model



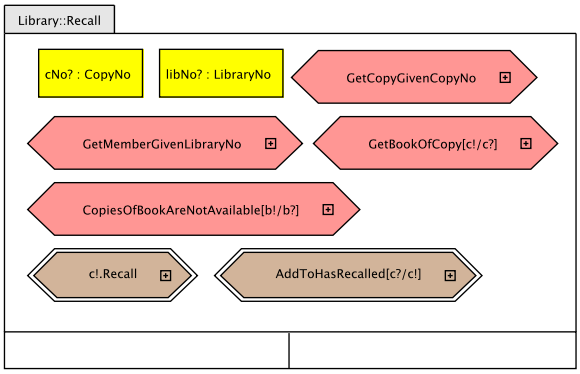
(a) Contract of operation `AddToBorrows`



(b) Contract of operation `BorrowsDelGivenCopy`

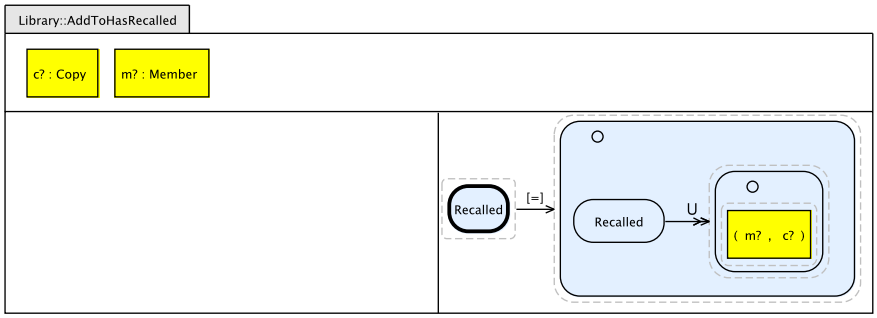


(c) Contract of operation `Renew`

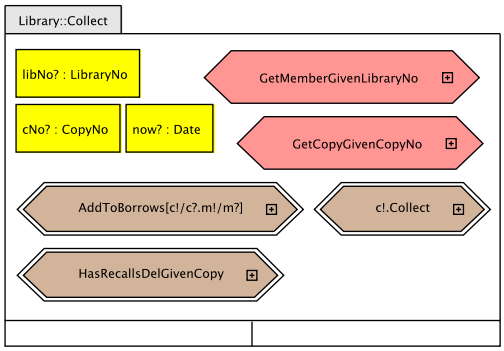


(d) Contract of operation `Recall`

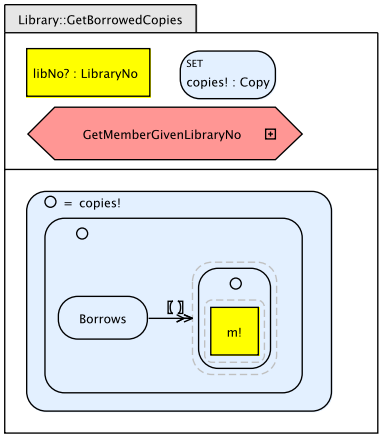
Figure C.21: Global Behavioural VCL model



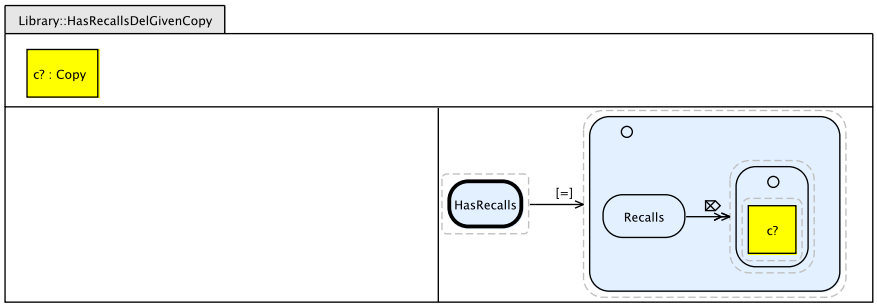
(a) Contract of operation AddToHasRecalled



(b) Contract of operation Collect

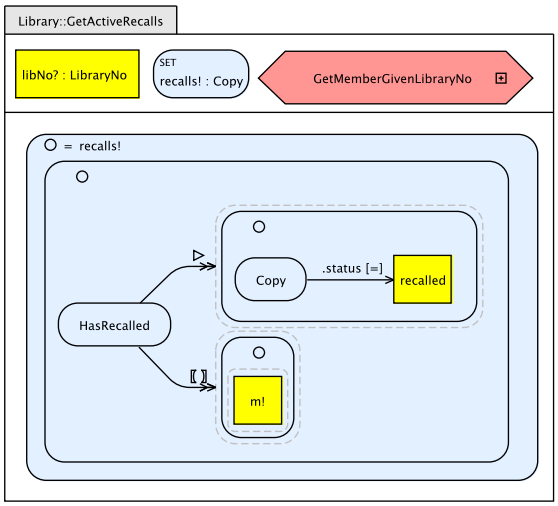


(c) AD of operation GetBorrowedCopies

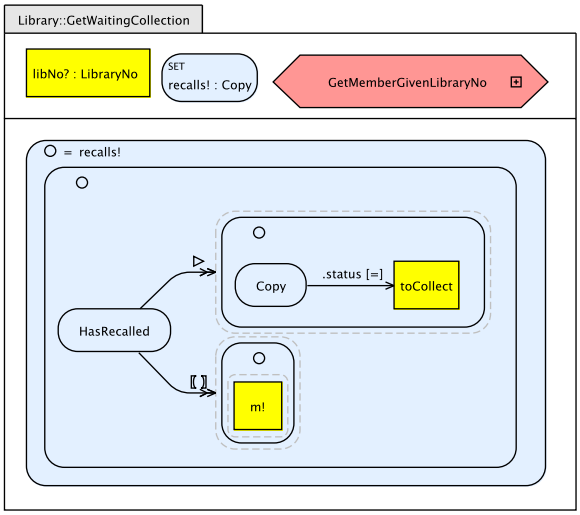


(d) Contract of operation HasRecallsDelGivenCopy

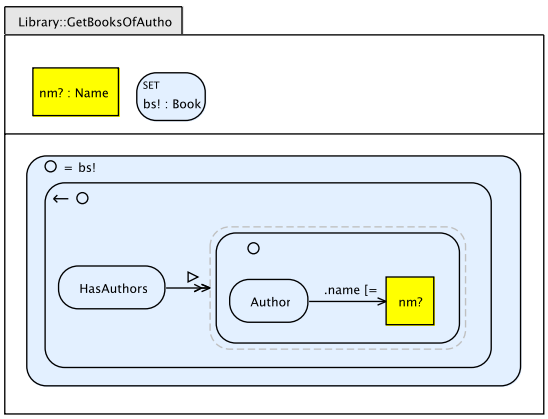
Figure C.22: Global Behavioural VCL model



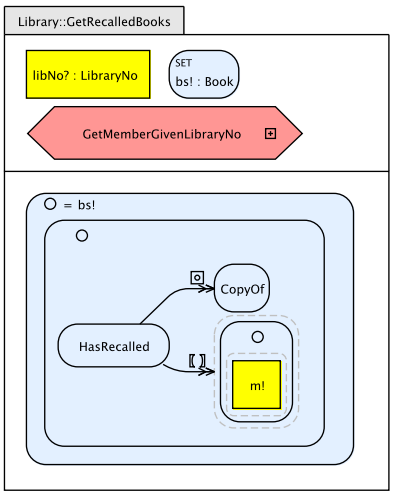
(a) AD of operation GetActiveRecalls



(b) AD of operation GetWaitingCollection

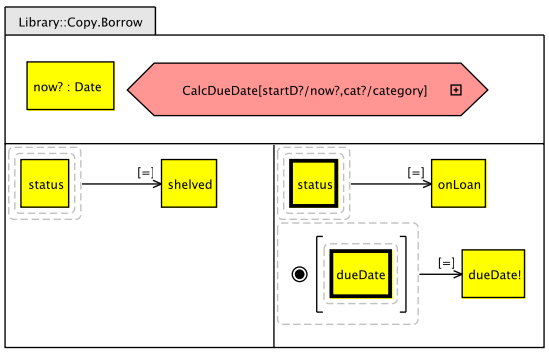


(c) AD of operation GetBooksOfAuthor

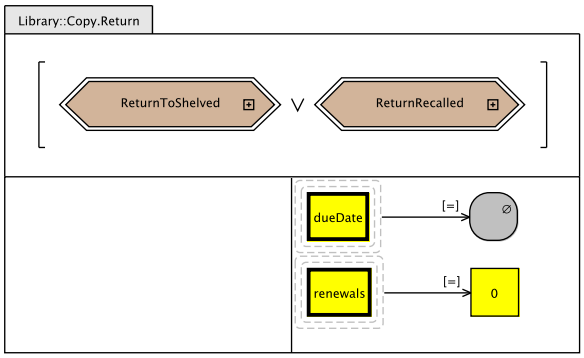


(d) AD of operation GetRecalledBooks

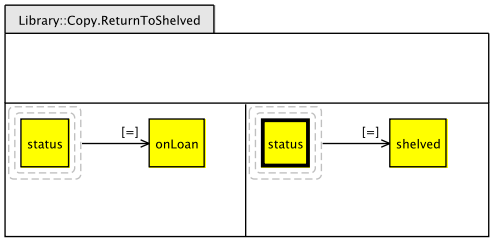
Figure C.23: Global Behavioural VCL model



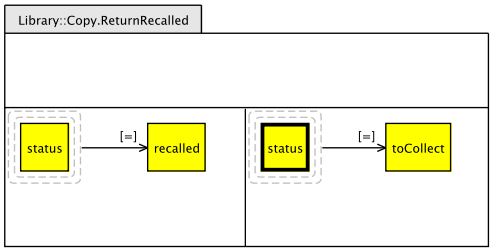
(a) CD of Copy.Borrow



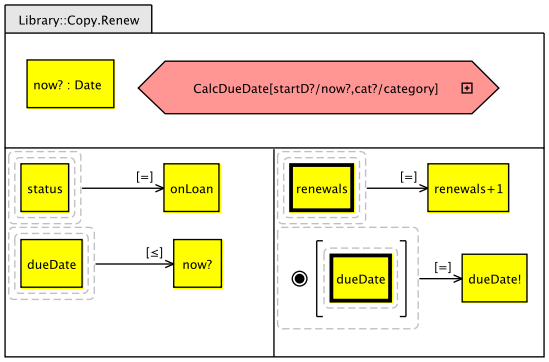
(b) CD of Copy.Return



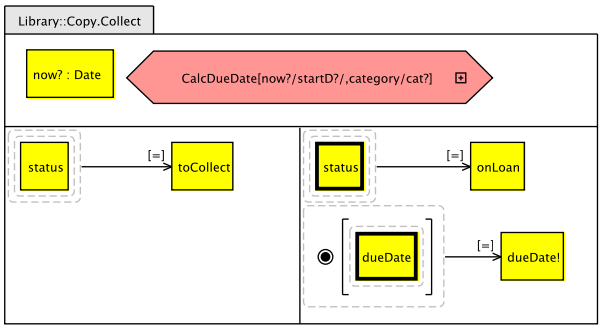
(c) CD of Copy.ReturnToShelved



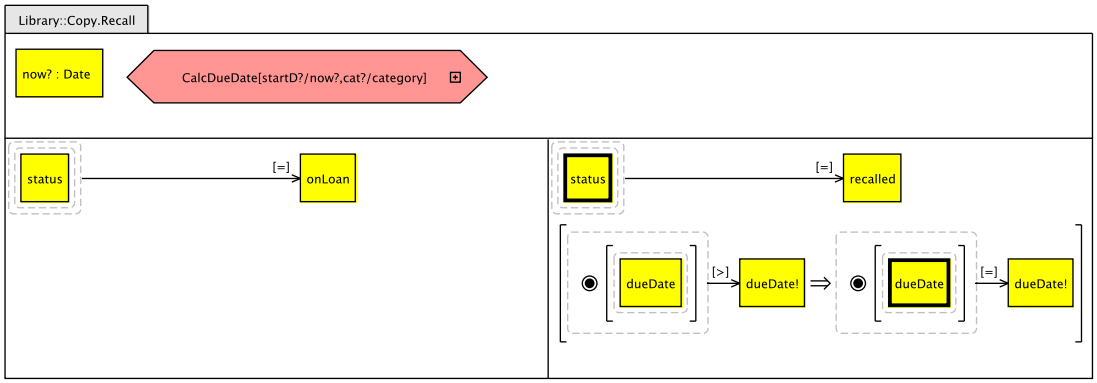
(d) CD of Copy.ReturnRecalled



(e) CD of Copy.Renew



(f) CD of Copy.Collect



(g) AD of Copy.Recall

Figure C.24: Local behaviour of set Copy in the VCL model

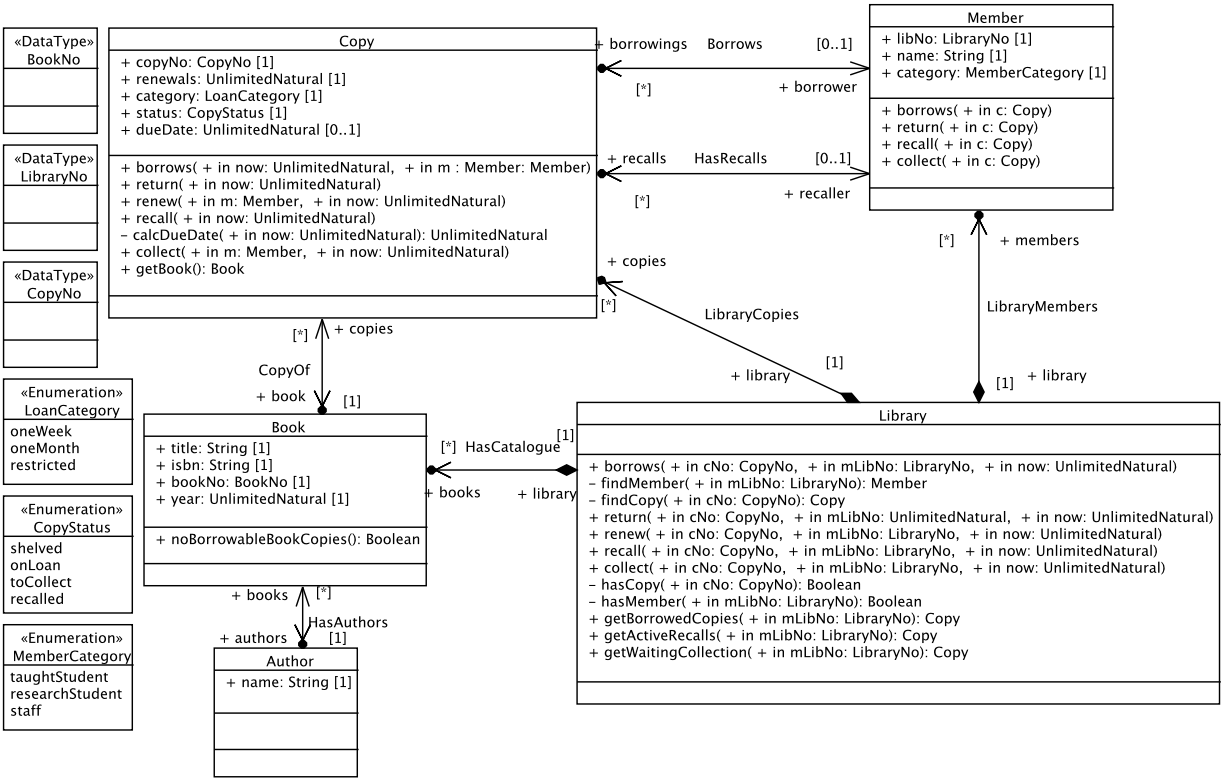


Figure C.25: Class diagram of the university library case study

```
context Library :: findMember(mLibNo : Library) : Member
post: result = members@pre->any(libNo = mLibNo)
```

(a) OCL operation *Library* :: *findMember*()

```
context Library :: findCopy(cNo : CopyNo) : Copy
post: result = copies@pre->any(copyNo=cNo)
```

(b) OCL operation *Library* :: *findCopy*()

```
context Library :: hasMember(mLibNo : Library) : Boolean
post: result = members->exists(m | m.libNo = mLibNo)
```

(c) OCL operation *Library* :: *hasMember*()

```
context Library :: hasCopy(cNo : CopyNo) : Boolean
post: result = copies->exists(c | c.copyNo = cNo)
```

(d) OCL operation *Library* :: *hasCopy*()

```
context Library::borrows(cNo : CopyNo, mLibNo : LibraryNo, now : UnlimitedNatural)
pre: hasCopy(cNo) and hasMember(mLibNo)
post:
  let c : Copy = findCopy(cNo), m : Member = findMember (mLibNo)
  in c.borrows(now, m) and m.borrows(c)
```

(e) OCL operation *Library* :: *borrows*()

```
context Library::return(cNo : CopyNo, mLibNo : LibraryNo, now : UnlimitedNatural)
pre: hasCopy(cNo) and hasMember(mLibNo)
post:
  let c : Copy = findCopy(cNo), m : Member = findMember (mLibNo)
  in c.return(now) and m.return(c)
```

(f) OCL operation *Library* :: *return*()

```
context Library::renew(cNo : CopyNo, mLibNo : LibraryNo, now : UnlimitedNatural)
pre: hasCopy(cNo) and hasMember(mLibNo)
post:
  let c : Copy = findCopy(cNo), m : Member = findMember (mLibNo)
  in c.renew(m, now)
```

(g) OCL operation *Library* :: *renew*()

```
context Library::recall(cNo : CopyNo, mLibNo : LibraryNo, now : UnlimitedNatural)
pre: hasCopy(cNo) and hasMember(mLibNo) and c.getBook()->noBorrowableBookCopies()
post:
  let c : Copy = findCopy(cNo), m : Member = findMember (mLibNo)
  in c.recall(m, now) and m.collect(c)
```

(h) OCL operation *Library* :: *recall*()

```
context Library::collect(cNo : CopyNo, mLibNo : LibraryNo, now : UnlimitedNatural)
pre: hasCopy(cNo) and hasMember(mLibNo)
post:
  let c : Copy = findCopy(cNo), m : Member = findMember (mLibNo)
  in c.collect(m, now) and m.collect(c)
```

(i) OCL operation *Library* :: *collect*()

Figure C.26: OCL operations of class `Library` in solution model of university library

```
context Member::borrows(c : Copy)
  post: borrowings = borrowings@pre()->union(Set{c})
```

(a) OCL operation *Member* :: *borrows*()

```
context Member::return(c : Copy)
  pre: borrowings->includes(c)
  post: borrowings = borrowings@pre()-Set{c}
```

(b) OCL operation *Member* :: *return*()

```
context Member::recall(c : Copy)
post: recalls = recalls@pre()->union(Set{c})
```

(c) OCL operation *Member* :: *recall*()

```
context Member::collect(c : Copy)
pre: recalls->includes(c)
post: recalls = recalls@pre() - Set{c}
  and borrowings = borrowings@pre()->union(Set{c})
```

(d) OCL operation *Member* :: *collect*()

```
context Book::noBorrowableBookCopies() : Boolean
post: result = copies->forAll(status<>CopyStatus::shelved or category = LoanCategory::restricted)
```

(e) OCL operation *Book* :: *noBorrowableBookCopies*()

Figure C.27: OCL operations of classes `Member` and `Book` in the model with seeded errors of the university library case study

```
context Copy::borrows(now : UnlimitedNatural, m : Member)
pre: status = CopyStatus::shelved
post: status = CopyStatus::onLoan and borrower = m and dueDate = calcDueDate(now)
```

(a) OCL operation *Copy* :: *borrow*()

```
context Copy::return(now : UnlimitedNatural)
pre: not borrower->notEmpty() and dueDate>= now
  and (status=CopyStatus::onLoan or status=CopyStatus::recalled)
post: renewals = 0 and dueDate->isEmpty() and borrower->isEmpty()
  and status@pre = onLoan implies status = CopyStatus::shelved
  and status@pre = recalled implies status = CopyStatus::toCollect
```

(b) OCL operation *Copy* :: *return*()

```
context Copy::renew(now : UnlimitedNatural, m : Member)
pre: status = CopyStatus::onLoan and borrower = m and dueDate <= now
post: renewals = renewals@pre()+1 and dueDate = calcDueDate (now)
```

(c) OCL operation *Copy* :: *renew*()

```
context Copy::recall(m : Member, now : UnlimitedNatural)
pre: status = CopyStatus::onLoan
post: status = CopyStatus::recalled
  and calcDueDate(now) < dueDate@pre implies dueDate = calcDueDate(now)
```

(d) OCL operation *Copy* :: *recall*()

```
context Copy::collect(m : Member, now : UnlimitedNatural)
pre: status = CopyStatus::toCollect and recaller = m
post: status = CopyStatus::onLoan and recaller->isEmpty()
  and borrower = m and dueDate = calcDueDate (now)
```

(e) OCL operation *Copy* :: *collect*()

```
context Copy::getBook() : Book
post: result = book
```

(f) OCL operation *Copy* :: *getBook*()

Figure C.28: OCL operations of class Copy in the solution model of university library

Appendix D

Flight Booking Materials

This chapter presents all the experiment materials that are related with the flight booking case study.

D.1 Starting Models

For the task on model construction of the state space, subjects were given partial models with some structures and invariants. The supplied models are as follows:

- The starting VCL model comprises an incomplete SD (Fig. D.1) and ADs of four invariants (Fig. D.2).
- The starting UML+OCL models comprises an incomplete class diagram and two OCL invariants (Fig. D.3)

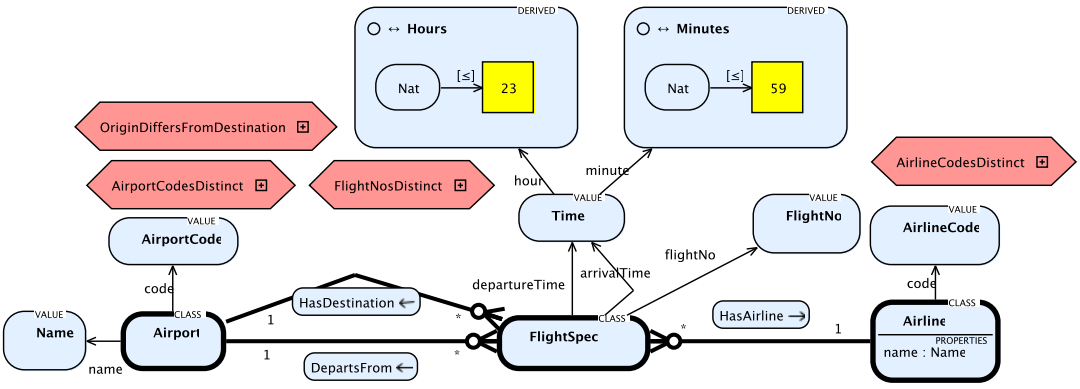
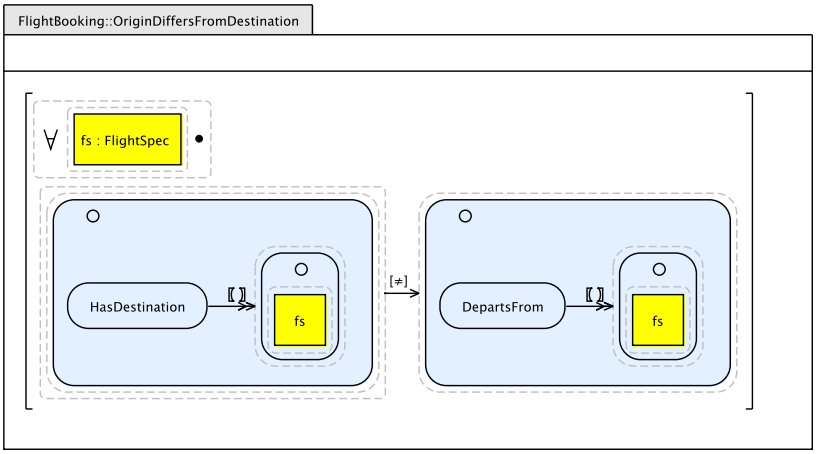
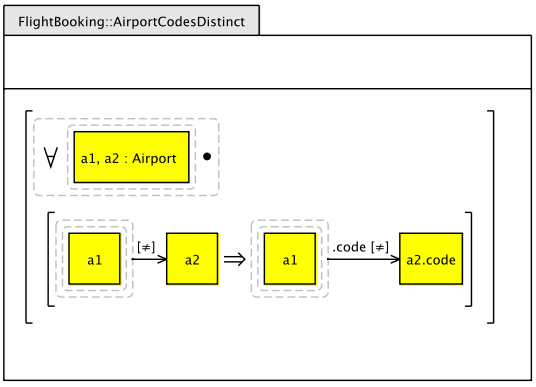


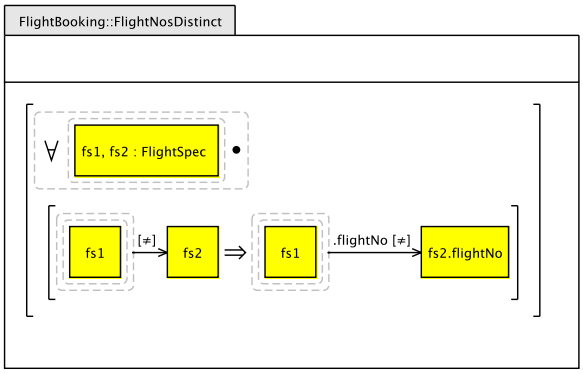
Figure D.1: The starting VCL model of flight booking



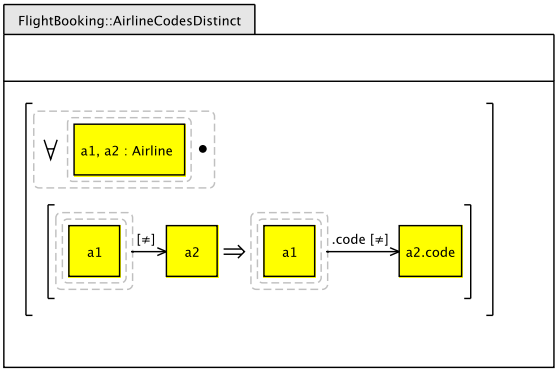
(a) AD of invariant OriginDiffersFromDestination



(b) AD of invariant AirportCodesDistinct

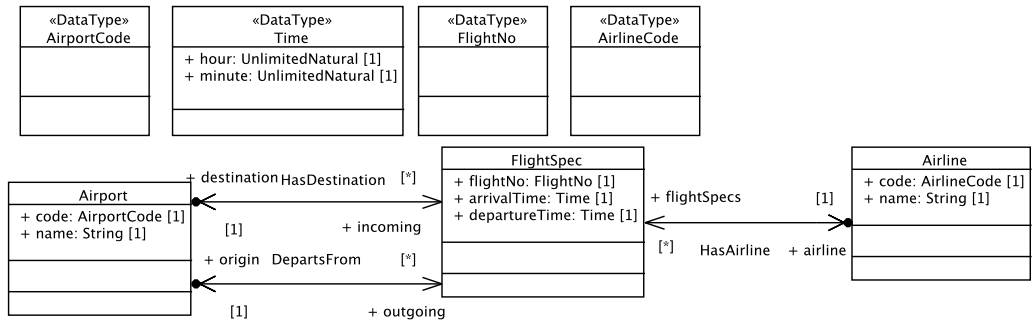


(c) AD of invariant FlightNosDistinct



(d) AD of invariant AirlineCodesDistinct

Figure D.2: VCL ADs of invariants of flight booking



(a) Class diagram

```
context Time
inv: self.hour <= 23 and self.minute <=59
```

(b) OCL invariant of UML datatype *Time*

```
context FlightSpec
inv OriginDiffersDestination:
  self.destination <> self.origin
```

(c) OCL invariant *OriginDiffersDestinations* of class *FlightSpec*

Figure D.3: Starting Model of flight booking

D.2 Intermediate Models

For the task on model construction of operations, subjects were given a complete state space model and a partial behavioural model and they were asked to model some operations of the system. The supplied models are as follows:

- The VCL state space model comprises the SD of Fig. D.4 and the ADs of Figs. D.2 and D.5. The partial VCL behavioural model comprises the BD of Fig. D.6 and the AD of Fig. D.7.
- The UML+OCL model comprises the class diagram of Fig. D.8, and the OCL invariants and operations of Fig. D.32.

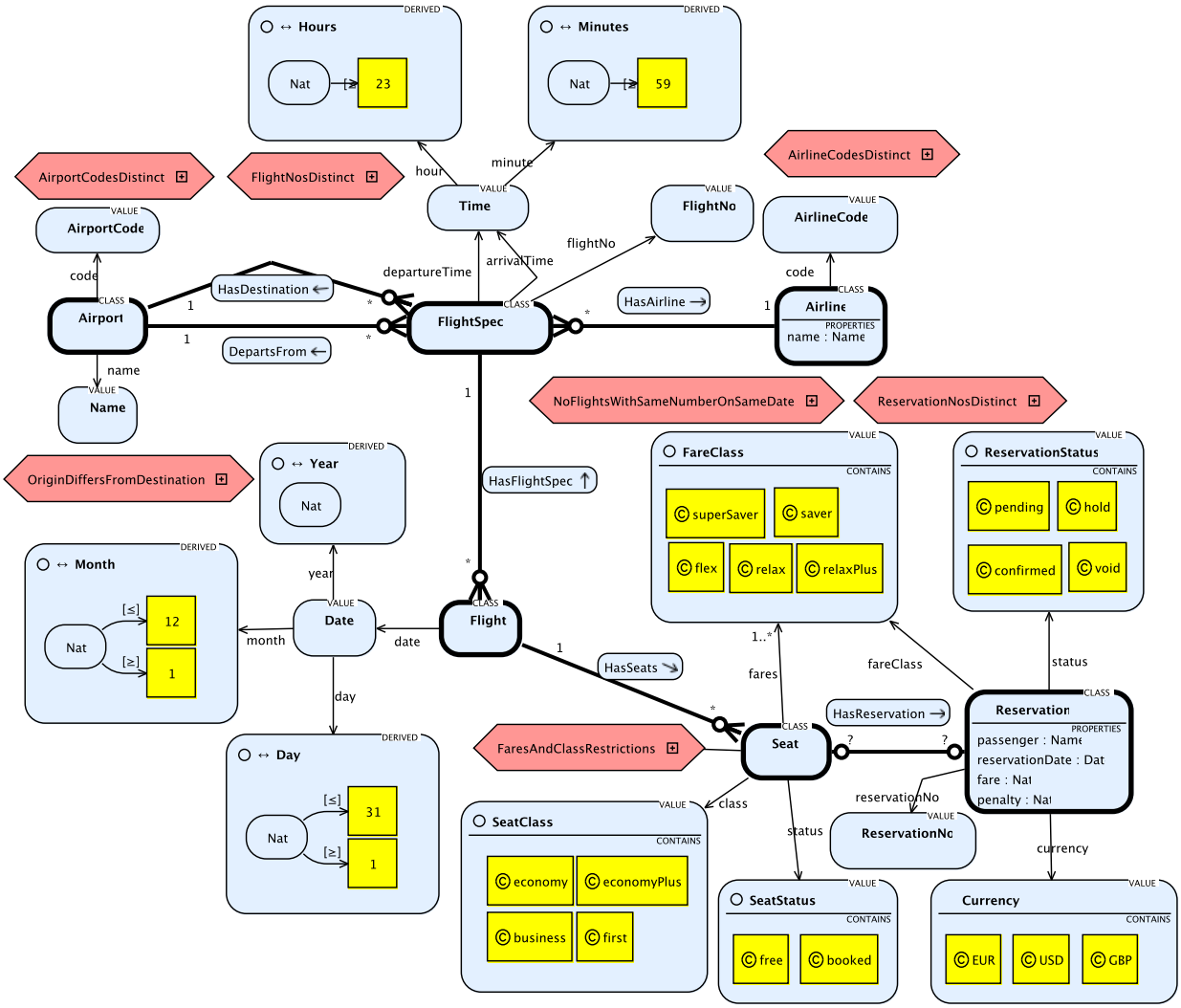
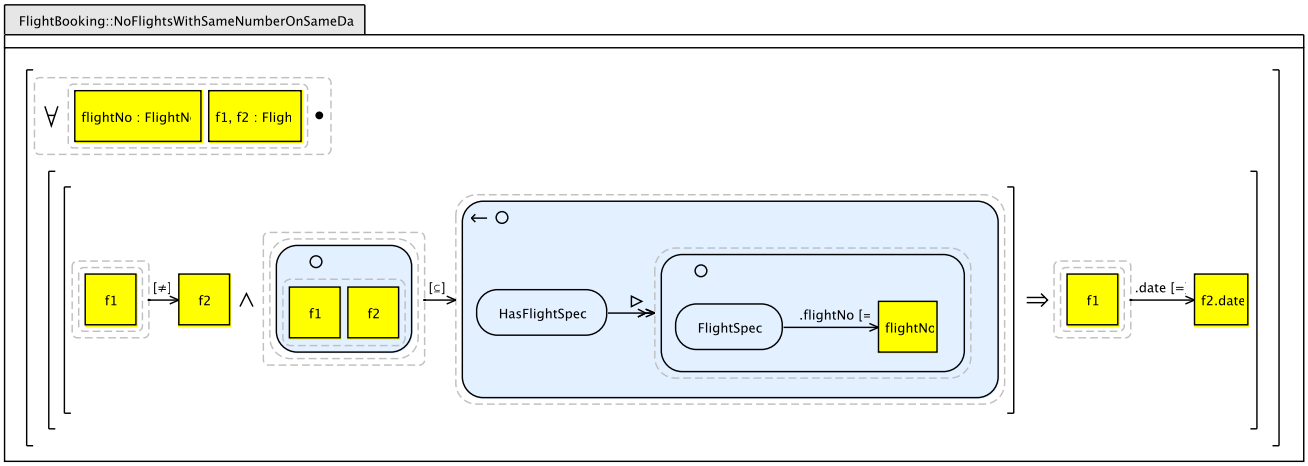
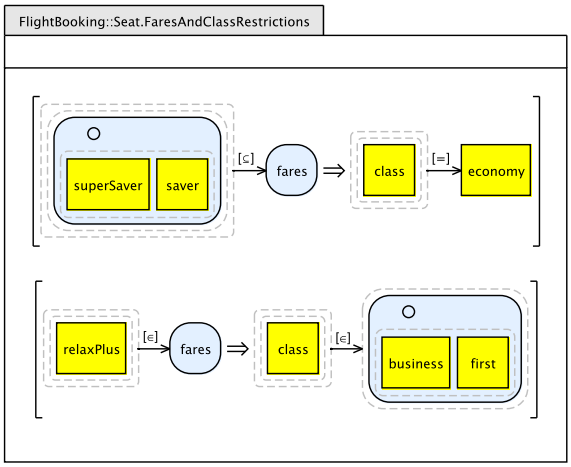


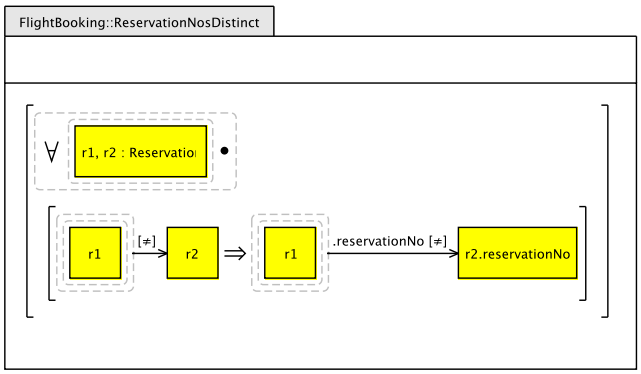
Figure D.4: The VCL SD of flight booking



(a) AD of invariant NoFlightsWithSameNumberOnSameDate



(b) AD of invariant Seat.FaresAndClassRestrictions



(c) AD of invariant ReservationNosDistinct

Figure D.5: ADs describing invariants of flight booking

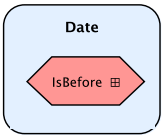


Figure D.6: VCL BD of the intermediate model of flight booking

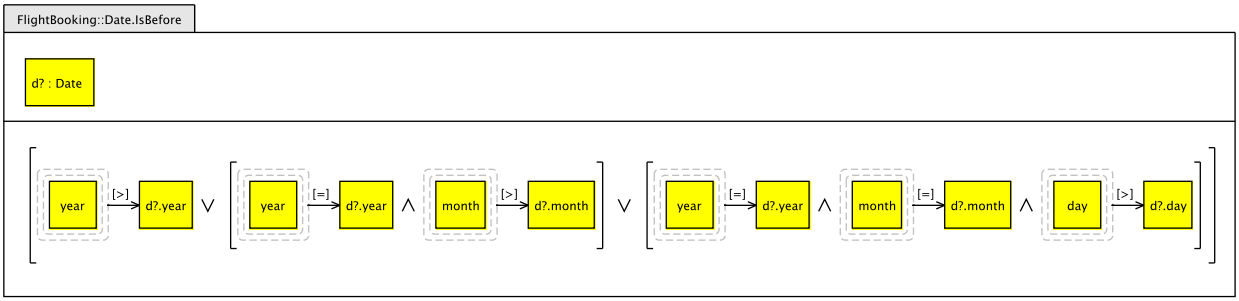


Figure D.7: VCL AD of operation `Date.IsBefore`

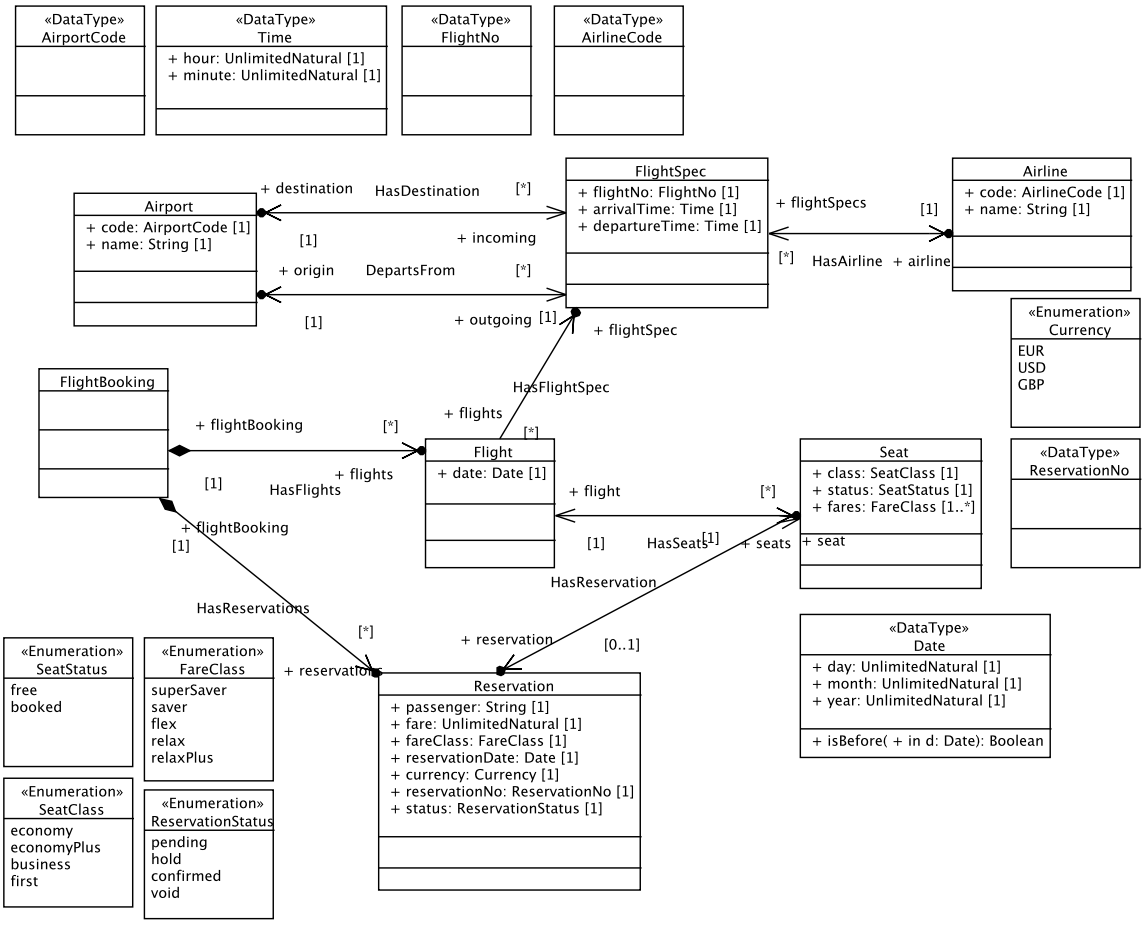


Figure D.8: Class diagram of the Flight Booking case study in the intermediate model

```
context FlightBooking
inv NoFlightsWithSameNoOnSameDate:
  flights->forAll(f1, f2 | f1 <> f2 and f1.flightSpec.flightNo = f2.flightSpec.flightNo
    implies f1.date <> f2.date)
```

(a) OCL invariant *NoFlightsWithSameNoOnSameDate* of class *FlightBooking*

```
context Seat
inv FaresAndClassRestrictions:
  fares->includesAll(Set {FareClass::superSaver, FareClass::saver})
    implies class = SeatClass::economy
  and fares->includes(FareClass::relaxPlus)
    implies class = SeatClass::business or class = SeatClass::first
```

(b) OCL invariant *Seat* :: *FaresAndClassRestrictions*

```
context Date
inv: day >= 1 and day <= 31 and month >=1 and month <=12
```

(c) OCL invariant of datatype *Date*

```
context Date::IsBefore (d : Date): Boolean
post:
result = self.year < d.year or (self.year = d.year and self.month < d.month)
  or (self.year = d.year and self.month = d.month and self.day < d.day)
```

(d) OCL operation *Date* :: *IsBefore*()

Figure D.9: OCL invariants and operations of flight Booking

D.3 Defect detection Models

The models with seeded errors for the defect detection task are as follows:

- The state space VCL model is given in Figs. D.10 to Fig. D.12. The behavioural VCL model is given in Figs. D.13 to D.18.
- The UML+OCL model is given in Figs. D.19 to D.22.

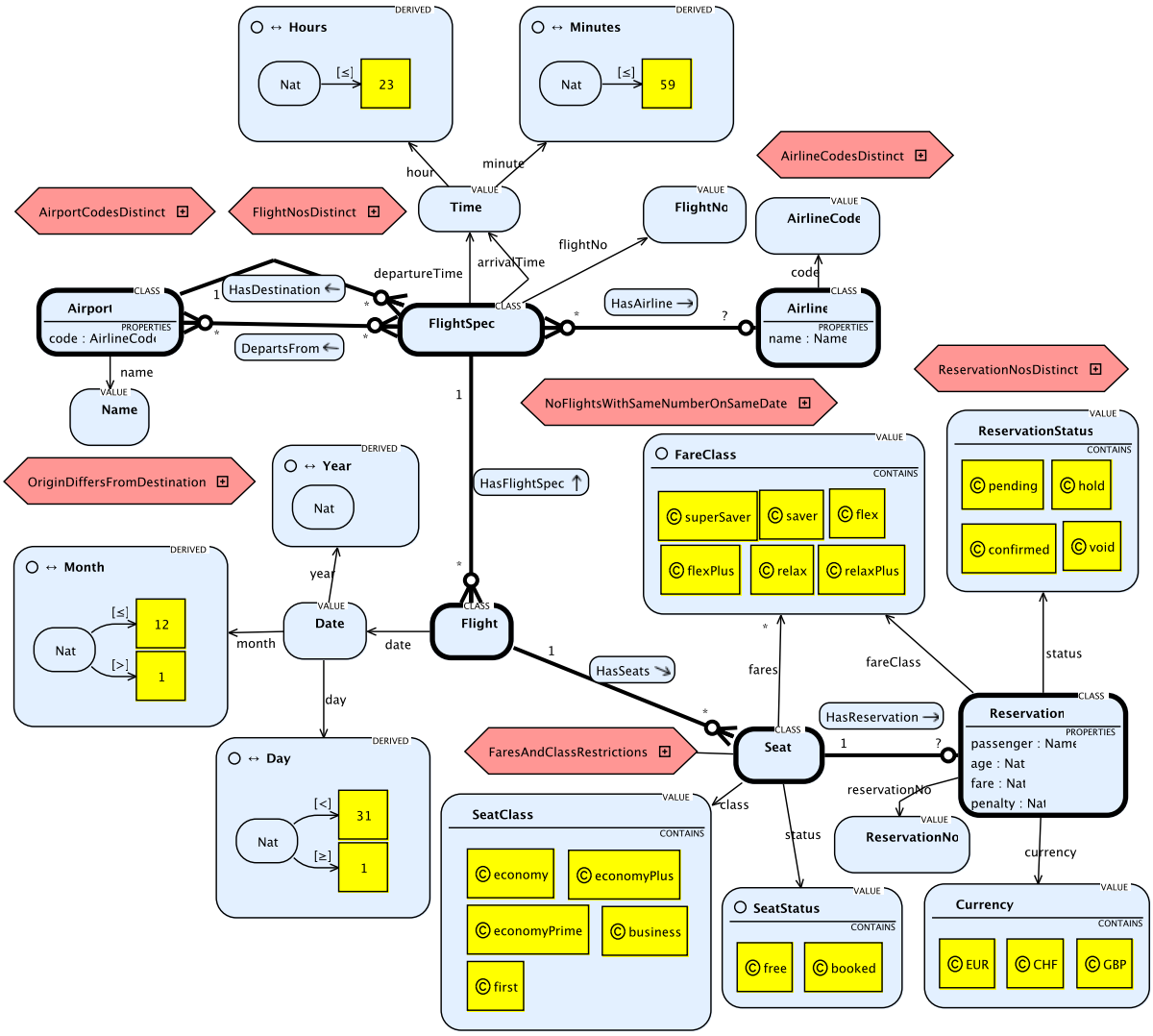
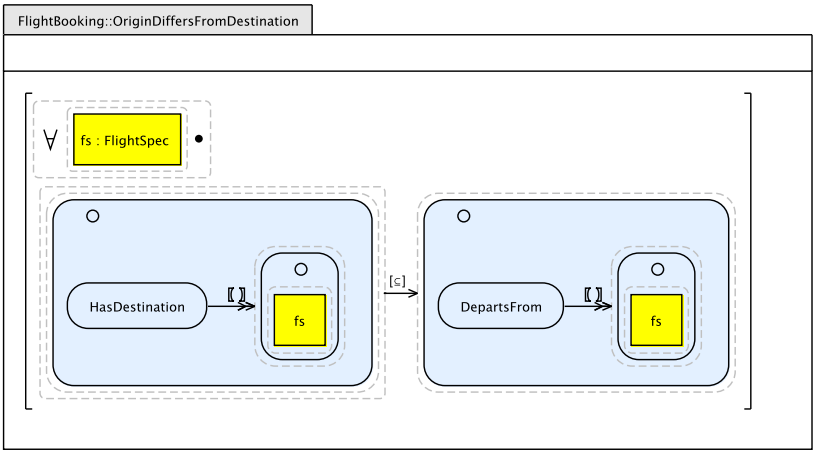
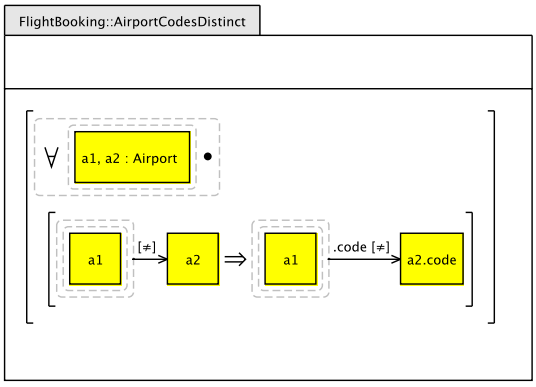


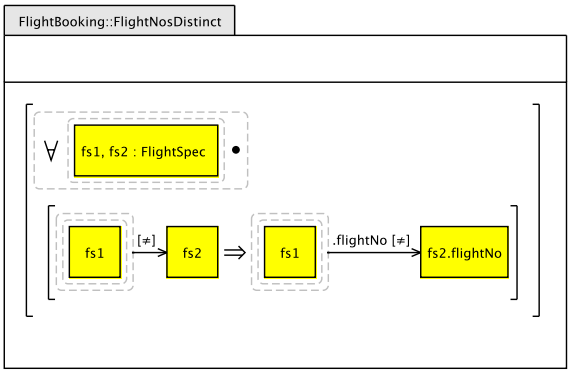
Figure D.10: VCL SD of flight booking in model with seeded errors



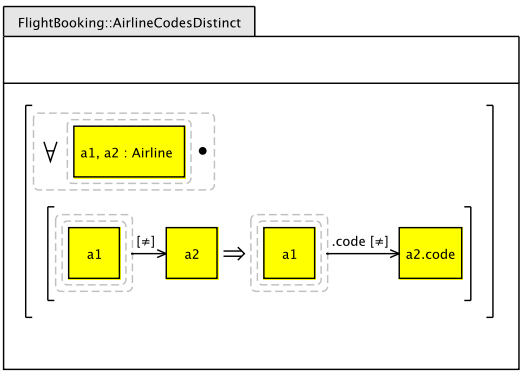
(a) AD of invariant OriginDiffersFromDestination



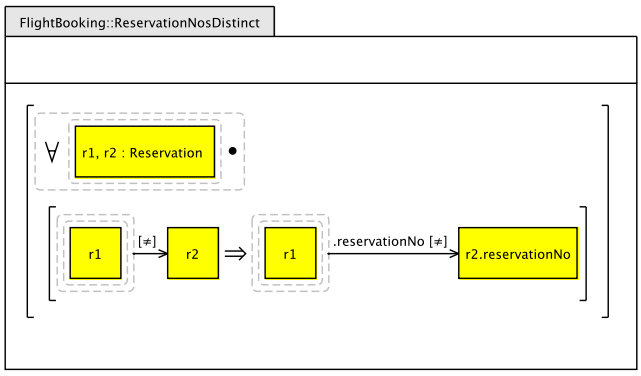
(b) AD of invariant AirportCodesDistinct



(c) AD of invariant FlightNosDistinct

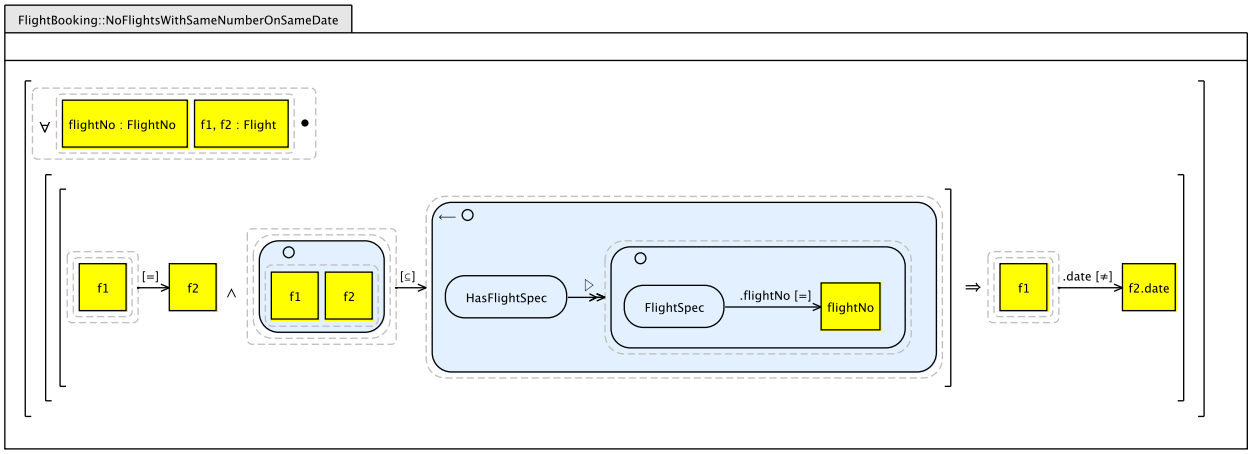


(d) AD of invariant AirlineCodesDistinct

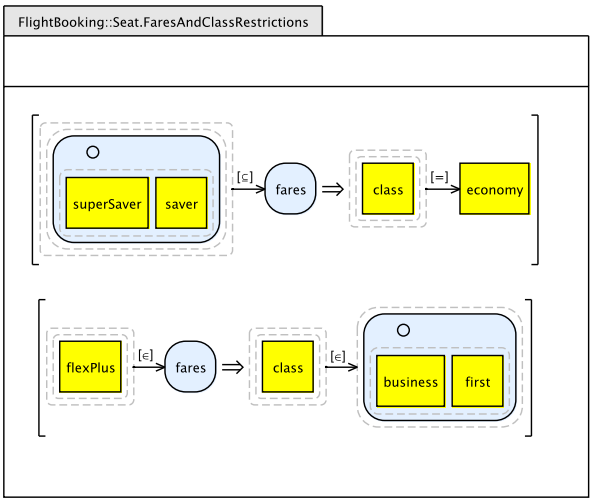


(e) AD of invariant ReservationNosDistinct

Figure D.11: VCL ADs of invariants of flight booking in model with seeded errors



(a) AD of invariant NoFlightsWithSameNumberOnSameDate



(b) AD of invariant Seat.FaresAndClassRestrictions

Figure D.12: VCL ADs describing invariants of flight booking in model with seeded errors

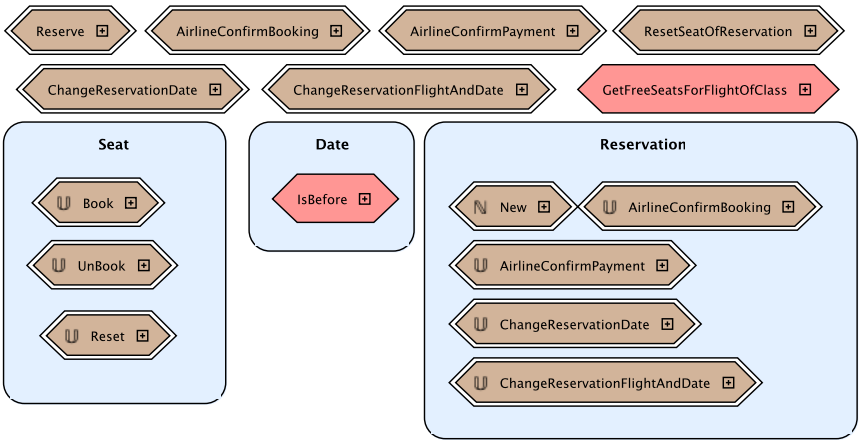
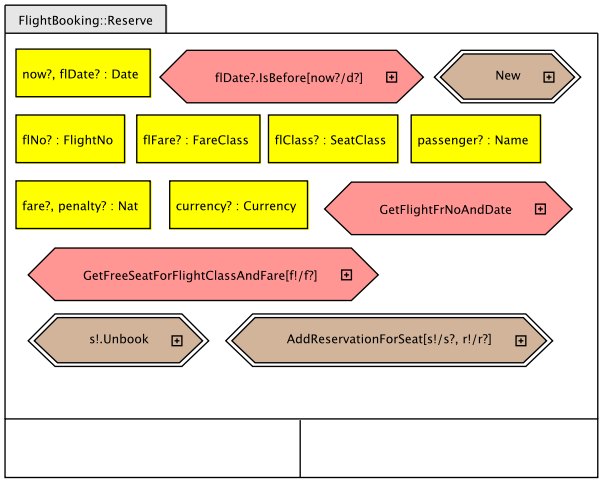
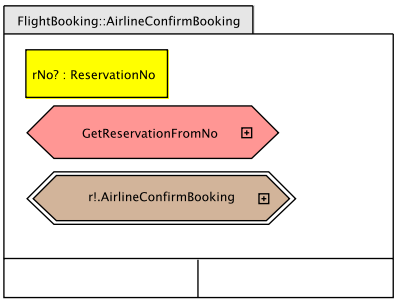


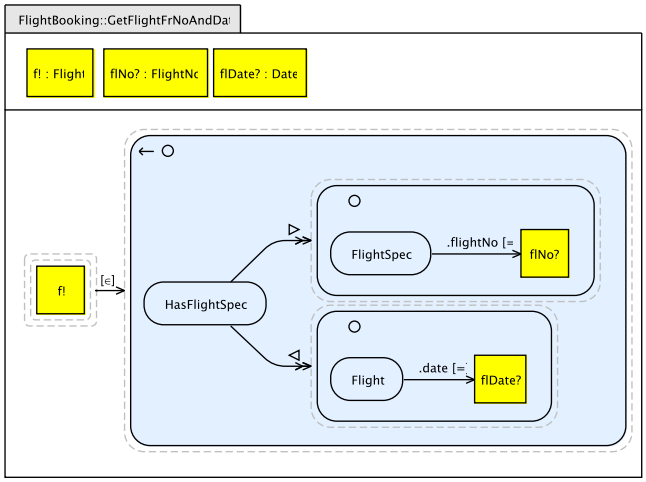
Figure D.13: VCL BD of flight booking in model with seeded errors



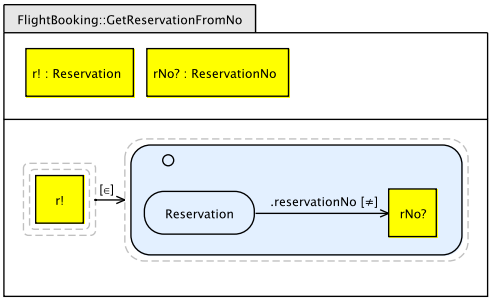
(a) Contract of operation **Reserve**



(b) Contract of operation **AirlineConfirmBooking**

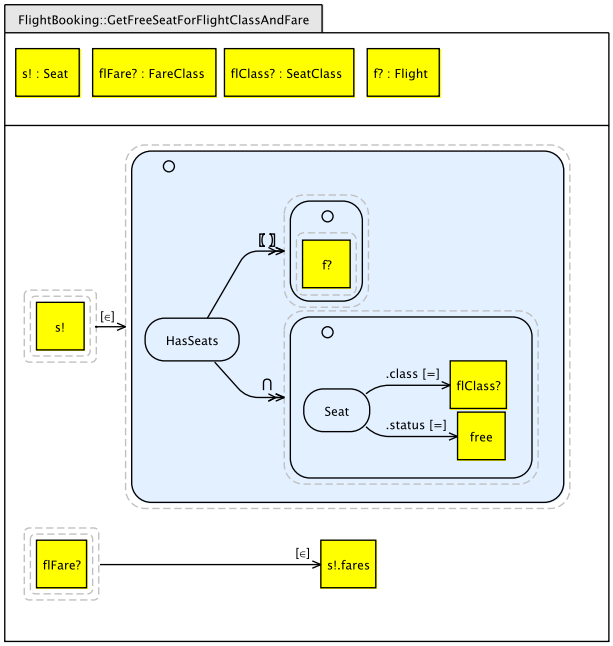


(c) AD of operation **GetFlightFrNoAndDate**

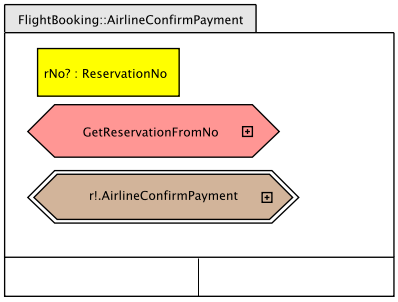


(d) AD of operation **GetReservationFromNo**

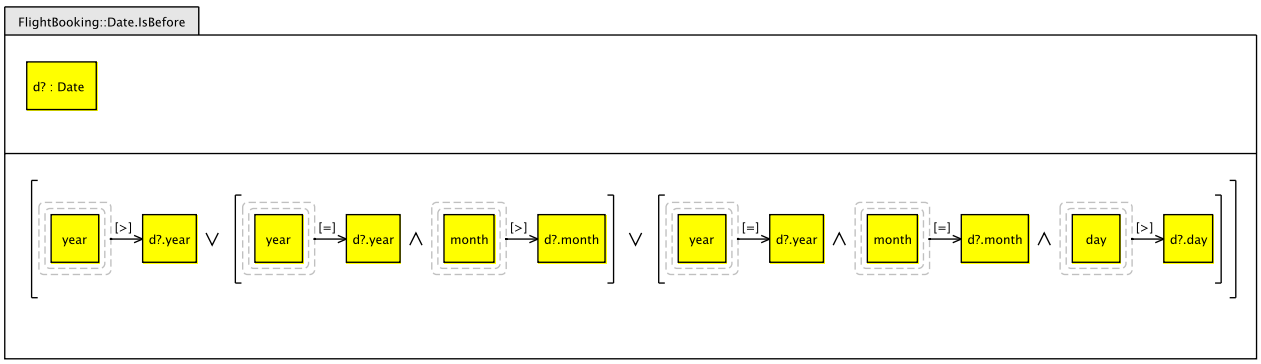
Figure D.14: VCL ADs and CDs describing operations of flight booking in model with seeded errors



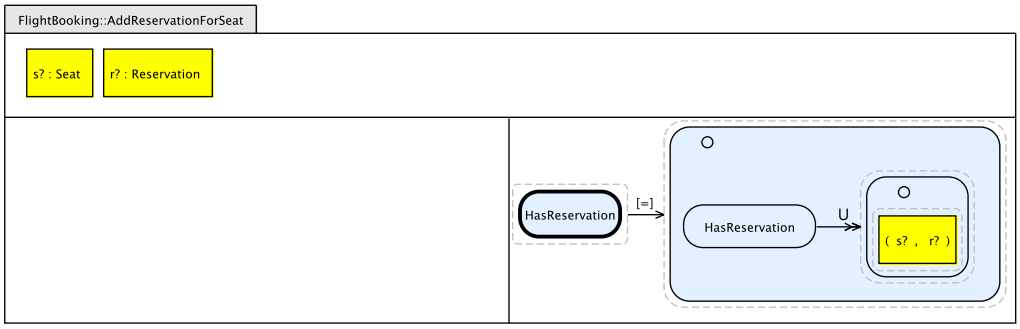
(a) AD of operation GetFreeSeatForFlightClassAndFare



(b) CD of operation AirlineConfirmPayment

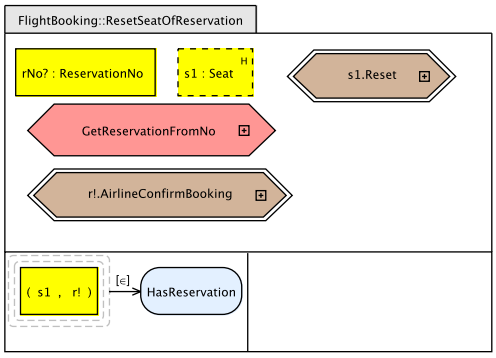


(c) AD of operation Date.IsBefore

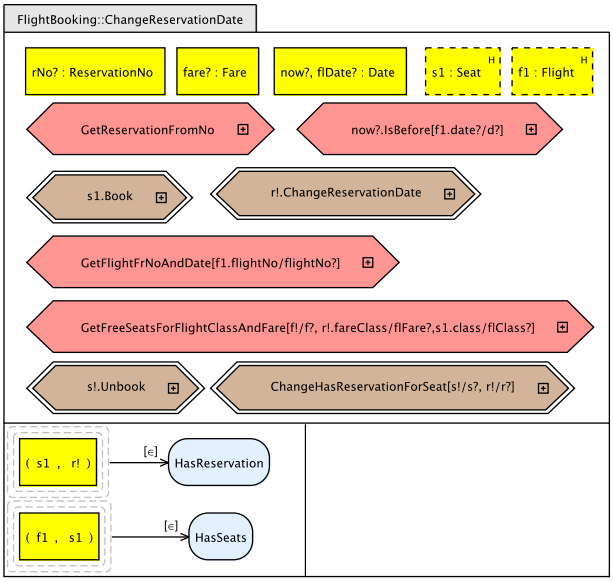


(d) CD of operation AddReservationForSeat

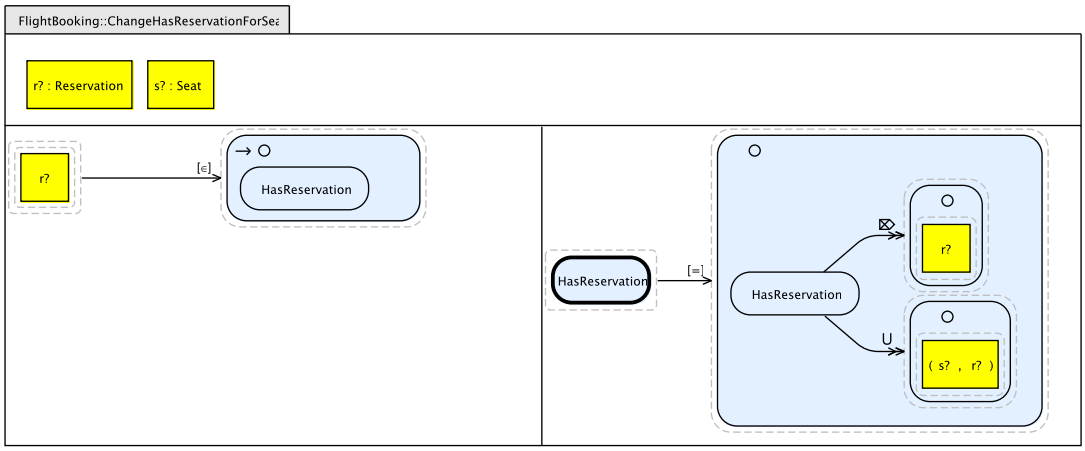
Figure D.15: VCL ADs and CDs describing operations of flight booking in model with seeded errors



(a) CD of operation ResetSeatOfReservation

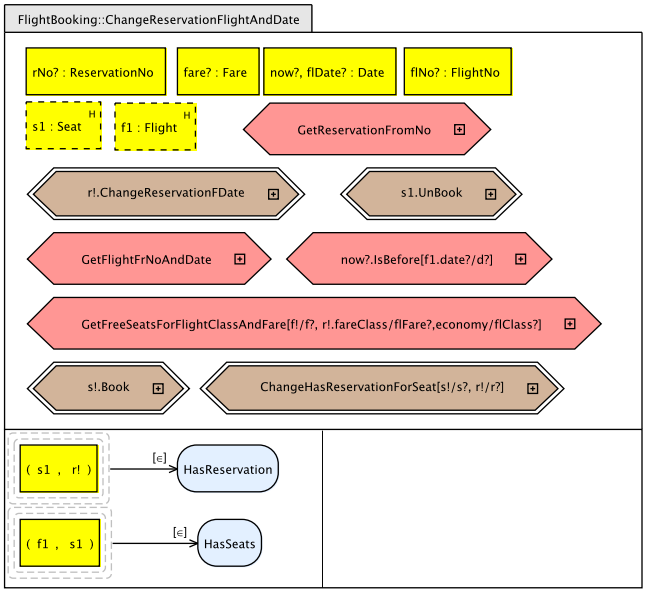


(b) CD of operation ChangeReservationDate

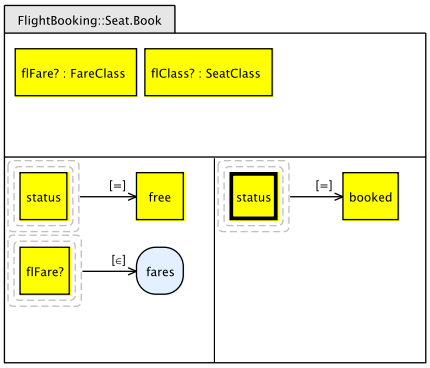


(c) CD of operation ChangeHasReservationForSeat

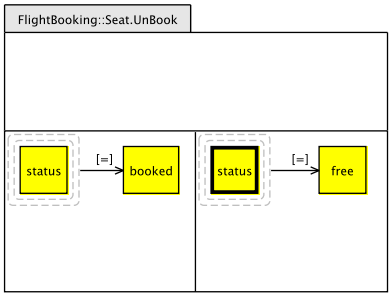
Figure D.16: VCL ADs and CDs describing operations of flight booking in model with seeded errors



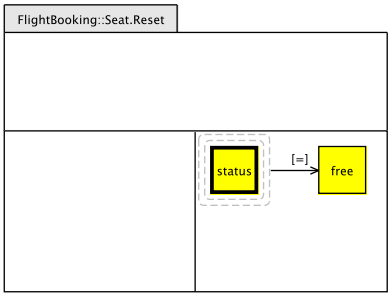
(a) CD of operation `ChangeReservationFlightAndDate`



(b) CD of operation `Seat.Book`

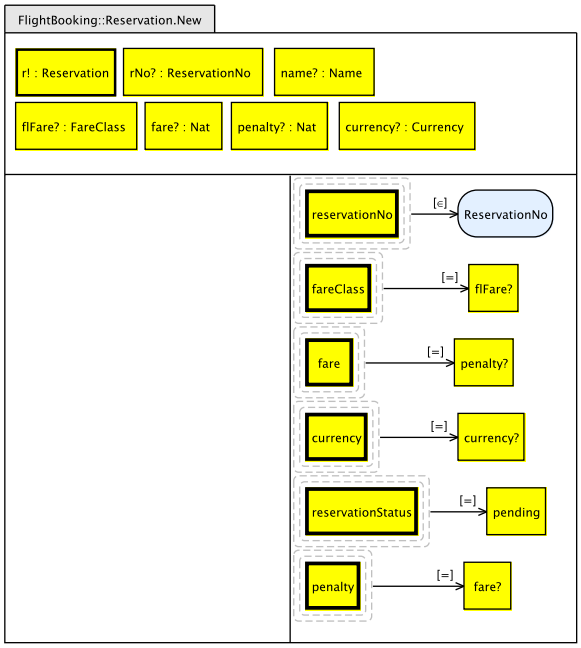


(c) CD of operation `Seat.UnBook`

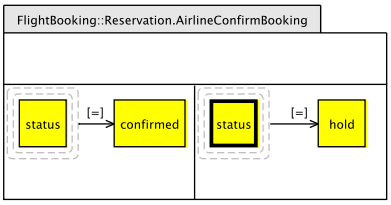


(d) CD of operation `Seat.Reset`

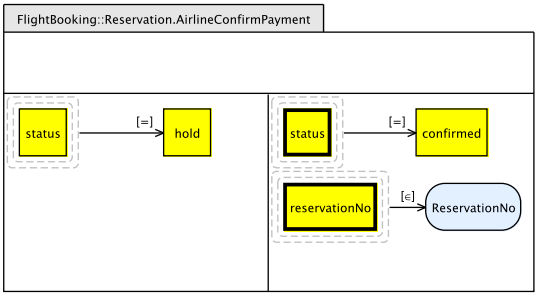
Figure D.17: VCL ADs and CDs describing operations of flight booking in model with seeded errors



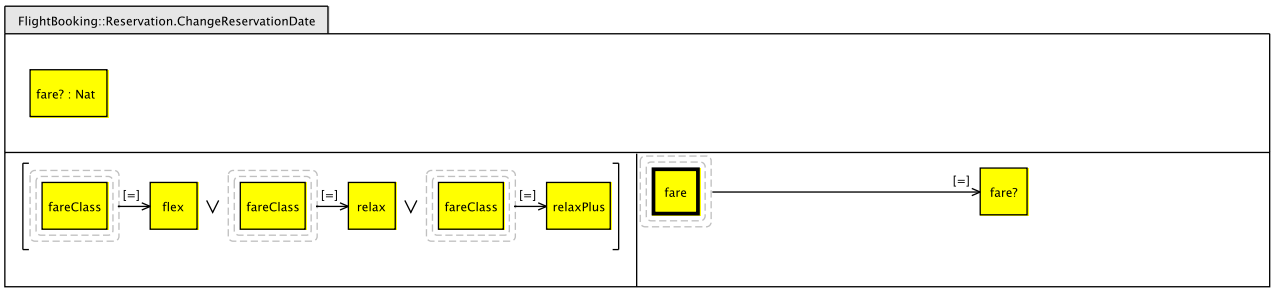
(a) CD of operation Reservation.New



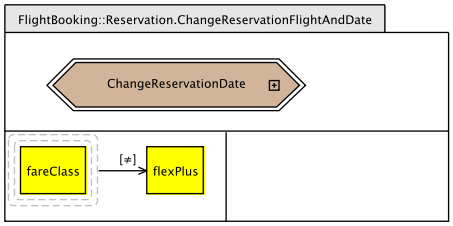
(b) CD of operation Reservation.AirlineConfirmBooking



(c) CD of operation Reservation.AirlineConfirmPayment



(d) CD of operation Reservation.ChangeReservationDate



(e) CD of operation Reservation.ChangeReservationFlightAndDate

Figure D.18: VCL ADs and CDs describing operations of flight booking in model with seeded errors

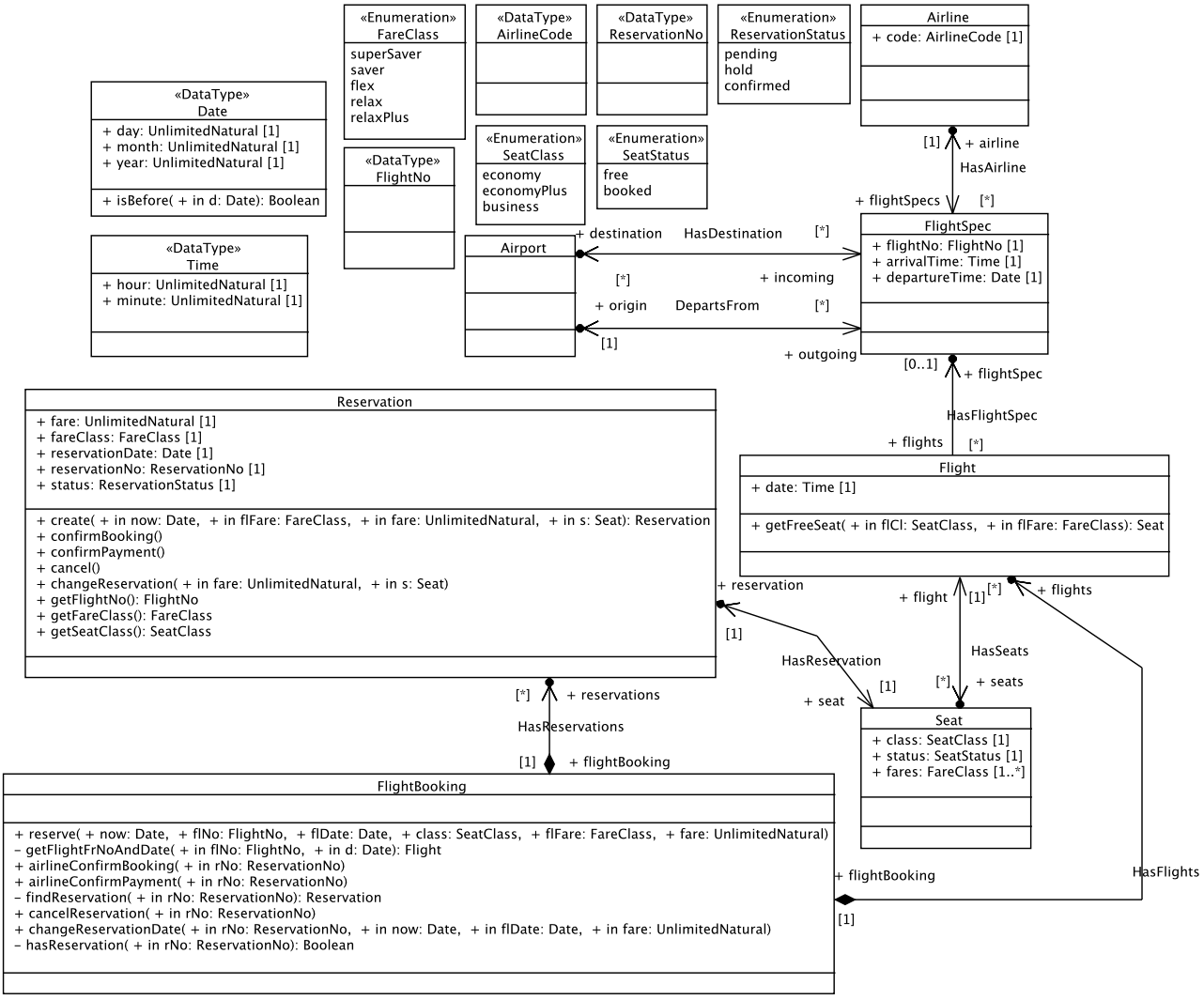


Figure D.19: UML Class diagram of flight booking in model with seeded errors

```
context Time
inv: self.hour <= 23 and self.minute <= 59
```

(a) OCL invariant of UML datatype *Time*

```
context FlightSpec
inv OriginDiffersDestination:
  self.destination <> self.origin
```

(b) OCL invariant *OriginDiffersDestinations* of class *FlightSpec*

```
context FlightBooking
inv NoFlightsWithSameNoOnSameDate:
flights->forAll(f1, f2 | f1 <> f2 and f1.flightSpec.flightNo = f2.flightSpec.flightNo
  implies f1.date <> f2.date)
```

(c) OCL invariant *NoFlightsWithSameNoOnSameDate* of class *FlightBooking*

```
context Seat
inv FaresAndClassRestrictions:
  fares->includesAll(Set {FareClass::superSaver, FareClass::saver})
    implies class = SeatClass::economy
  and fares->includes(FareClass::relaxPlus)
    implies class <> SeatClass::business and class <> SeatClass::first
```

(d) OCL invariant *Seat* :: *FaresAndClassRestrictions*

```
context Date
inv: day >= 1 and day < 31 and month >=1 and month <=12
```

(e) OCL invariant of datatype *Date*

```
context Date::IsBefore (d : Date): Boolean
post:
result = self.year < d.year or (self.year = d.year and self.month < d.month)
  or (self.year = d.year and self.month = d.month and self.day < d.day)
```

(f) OCL operation *Date* :: *IsBefore*()

Figure D.20: OCL invariants and operations of flight Booking in model with seeded errors

```
context FlightBooking getFlightFrNoAndDate(flNo : FlightNo, d : Date) : Flight
post: result = self.flights->select(f | f.flightSpec.flightNo = flNo and f.date = d)->any(true)
```

(a) OCL operation *FlightBooking* :: *getFlightFrNoAndDate*()

```
context FlightBooking reserve(now : Date, flNo : FlightNo, flDate : Date,
  class : SeatClass, flFare : FareClass, fare : UnlimitedNatural)
pre: now.isBefore(flDate)
post: let f = getFlightFrNoAndDate(flNo, flDate), s = f.getFreeSeat(class, flFare)
  in Reservation::create(now, flFare, fare, s)
```

(b) OCL operation *FlightBooking* :: *reserve*()

```
context FlightBooking hasReservation(rNo : ReservationNo) : Boolean
post: reservations->exists(r | r.reservationNo = rNo)
```

(c) OCL operation *FlightBooking* :: *hasReservation*()

```
context FlightBooking findReservation(rNo : ReservationNo) : Reservation
pre: hasReservation(rNo)
post: result = reservations->any(reservationNo = rNo)
```

(d) OCL operation *FlightBooking* :: *findReservation*()

```
context FlightBooking airlineConfirmBooking(rNo : ReservationNo)
pre: hasReservation(rNo)
post: let r = findReservation (rNo)
  in r.confirmPayment()
```

(e) OCL operation *FlightBooking* :: *airlineConfirmBooking*()

```
context FlightBooking airlineConfirmPayment(rNo : ReservationNo)
pre: hasReservation(rNo)
post: let r = findReservation
  in r.confirmPayment()
```

(f) OCL operation *FlightBooking* :: *airlineConfirmPayment*()

```
context FlightBooking cancelReservation(rNo : ReservationNo)
pre: hasReservation(rNo)
post: let r = findReservation(rNo)
  in r.cancel() and reservations->excludes(r)
```

(g) OCL operation *FlightBooking* :: *cancelReservation*()

```
context FlightBooking changeReservationDate(rNo : ReservationNo, now : Date,
  flDate : Date, fare: UnlimitedNatural)
pre: hasReservation(rNo) and now.isBefore(flDate)
post: let r = findReservation(rNo),
      f = getFlightFrNoAndDate (r.getFlightNo(), flDate),
      s = f.getFreeSeat(r.getClassSeat(), r.getFareClass())
      in r.changeReservation(fare, s)
```

(h) OCL operation *FlightBooking* :: *changeReservationDate*()

Figure D.21: OCL operations of flight booking in model with seeded errors

```
context Flight getFreeSeat(flCl : SeatClass, flFare : FareClass) : Seat
post: result = self.seats->select (s | s.class = flCl and s.fares->includes(flFare))->any(true)
```

(a) OCL operation *Flight* :: *getFreeSeat*()

```
context Reservation
  create(now: Date, flFare : FareClass, fare : UnlimtedNatural, s : Seat) : Reservation
post: self.reservationNo = ReservationNo.allInstances->any(true)
  and self.fareClass = flFare and self.reservationDate = now
  and self.currency = currency and self.fare = fare
  and seat = s and status = ReservationStatus::hold
  and result = self
```

(b) OCL operation *Reservation* :: *create*()

```
context Reservation confirmBooking ()
pre: self.status = ReservationStatus::hold
post: self.status = ReservationStatus::pending
```

(c) OCL operation *Reservation* :: *confirmBooking*()

```
context Reservation confirmPayment ()
pre: self.status = ReservationStatus::pending
post: self.status = ReservationStatus::confirmed
```

(d) OCL operation *Reservation* :: *confirmPayment*()

```
context Reservation changeReservation ()
pre: (fareClass  = FareClass::flex or fareClass  = FareClass::relax
    or fareClass  = FareClass::relaxPlus)
  and status <> confirmed
post: self.fare <> fare and seat->isEmpty()
```

(e) OCL operation *Reservation* :: *changeReservation*()

```
context Reservation getFlightNo () : FlightNo
post: result = self.seat.flight.flightSpec.flightNo
```

(f) OCL operation *Reservation* :: *getFlightNo*()

```
context Reservation getFareClass () : FareClass
post: result = seat.fares->any(true)
```

(g) OCL operation *Reservation* :: *getFareClass*()

```
context Reservation getSeatClass () : SeatClass
post: result = self.seat.class
```

(h) OCL operation *Reservation* :: *getSeatClass*()

Figure D.22: OCL operations of flight booking in model with seeded errors

D.4 Solution Models

The solution models were given to subjects as part of the model comprehension task. They are as follows:

- The VCL solution model consists of the state space model presented in Figs. D.4, D.2 and D.5, and the behavioural model of Figs. D.23 to D.30
- The UML+OCL solution model is presented in Figs. D.31 to D.34.

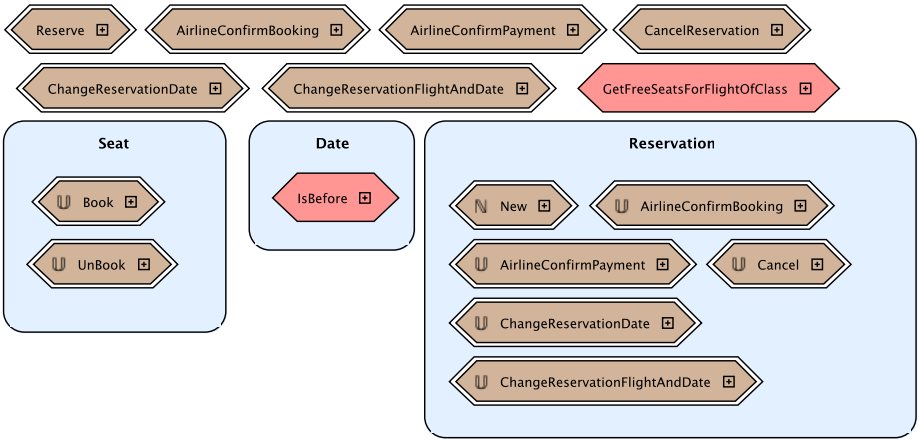
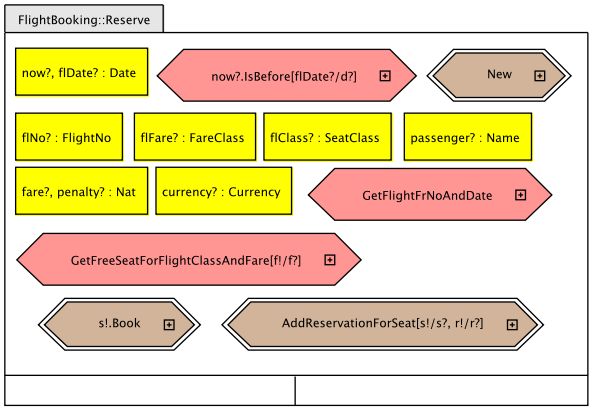
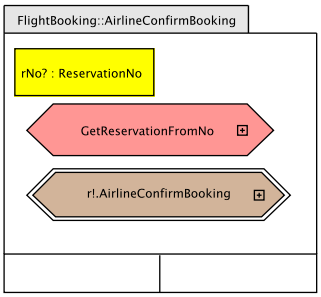


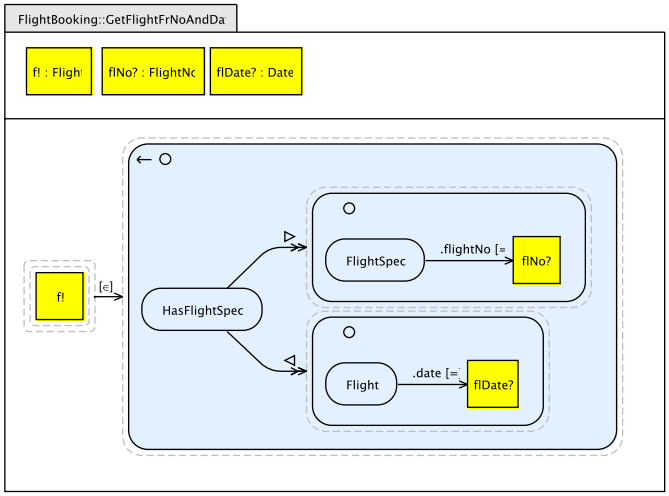
Figure D.23: VCL BD of flight booking



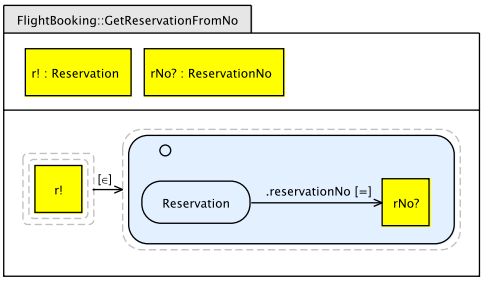
(a) CD of operation Reserve



(b) CD of operation AirlineConfirmBooking

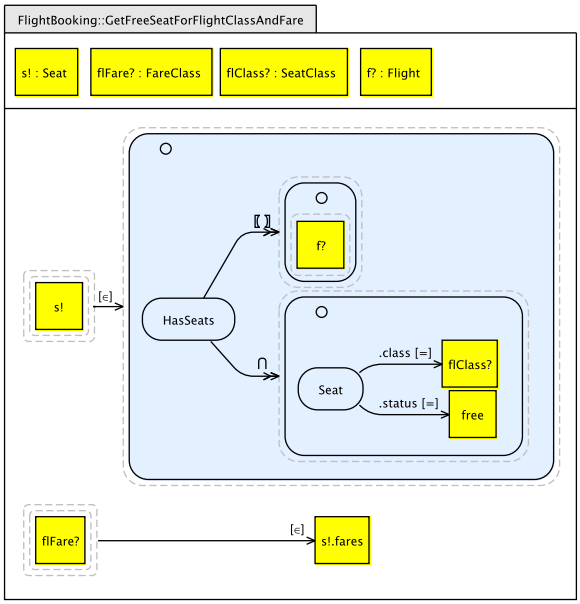


(c) AD of operation GetFlightFrNoAndDate

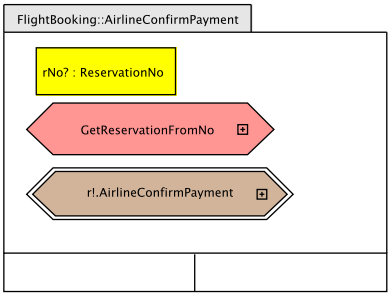


(d) AD of operation GetReservationFromNo

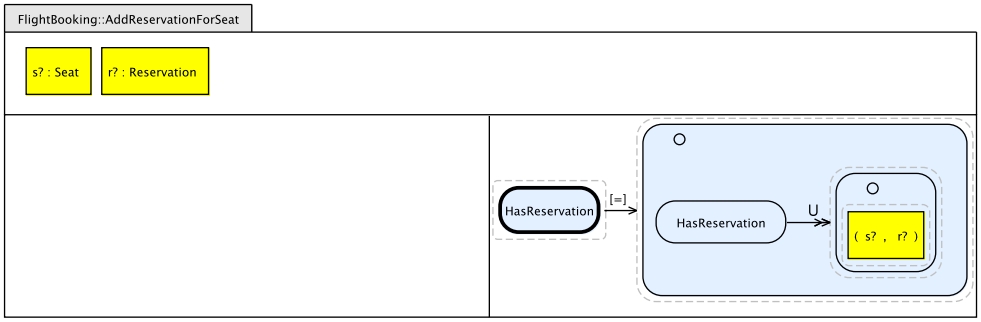
Figure D.24: VCL CDs and ADs of the behavioural model of flight booking



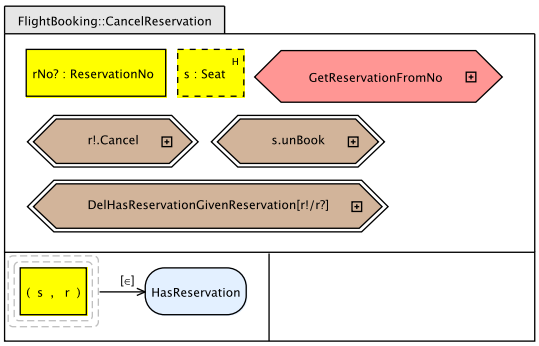
(a) AD of operation GetFreeSeatForFlightClassAndFare



(b) CD of operation AirlineConfirmPayment

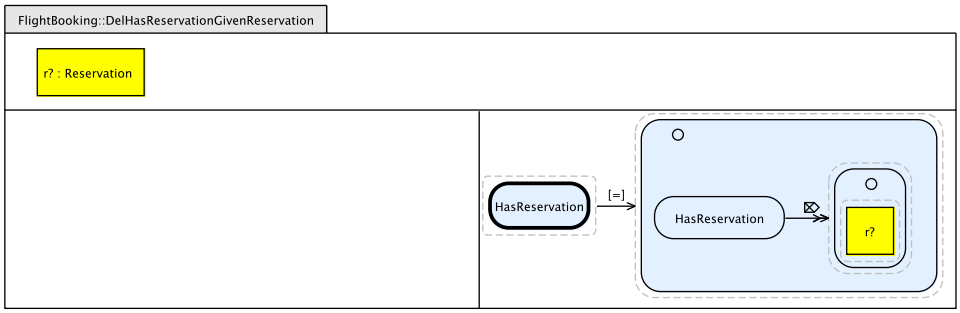


(c) CD of operation AddReservationForSeat

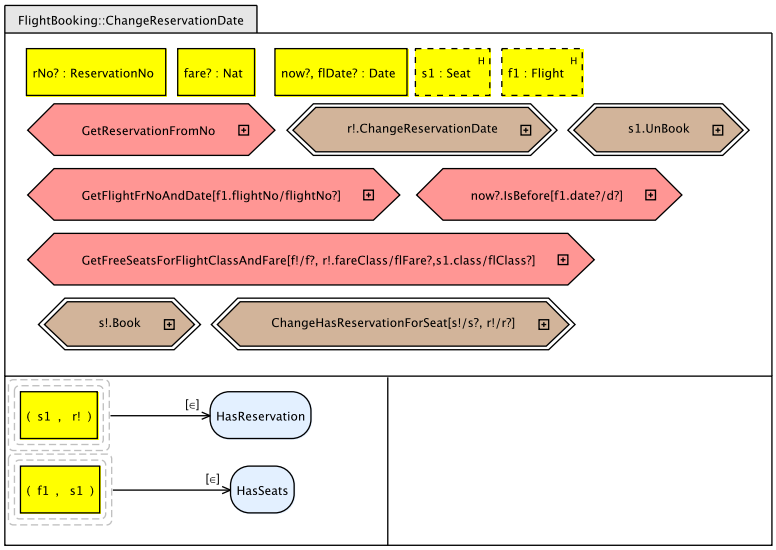


(d) CD of operation CancelReservation

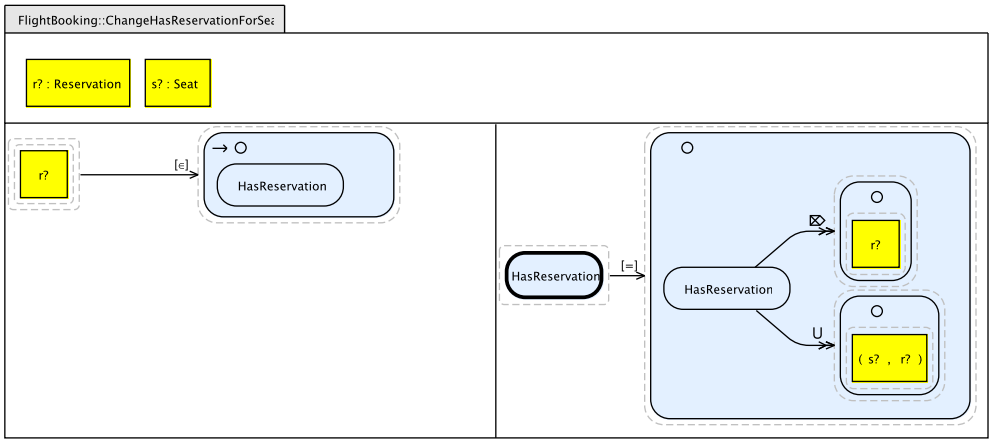
Figure D.25: VCL CDs and ADs of the behavioural model of flight booking



(a) CD of operation DelHasReservationGivenReservation

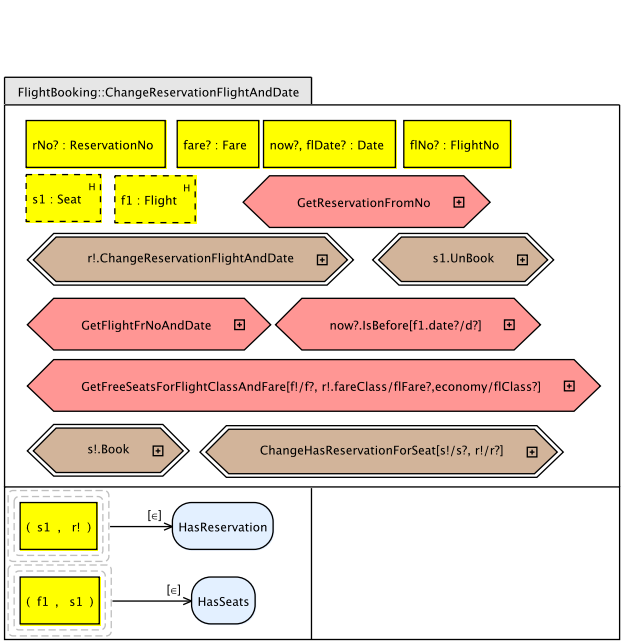


(b) CD of operation ChangeReservationDate

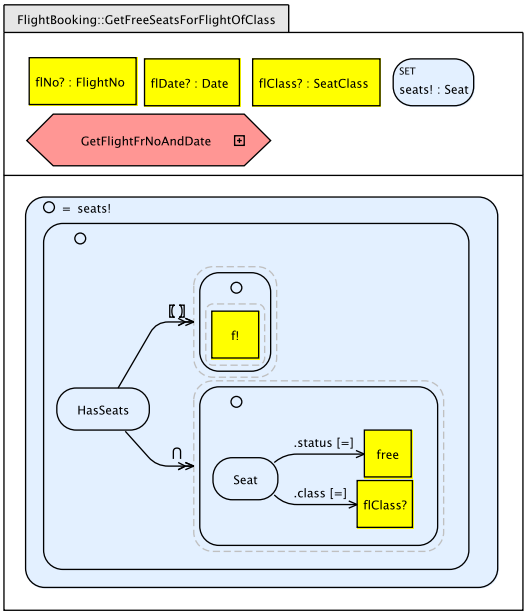


(c) CD of operation ChangeHasReservationForSeat

Figure D.26: VCL CDs and ADs of the behavioural model of flight booking

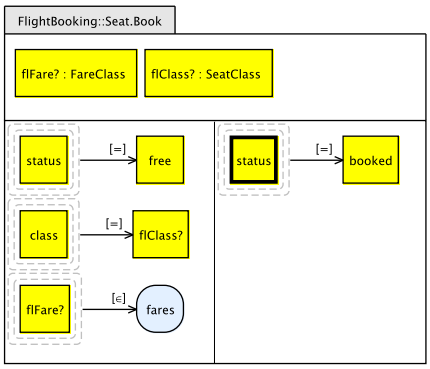


(a) CD of operation `ChangeReservationFlightAndDate`

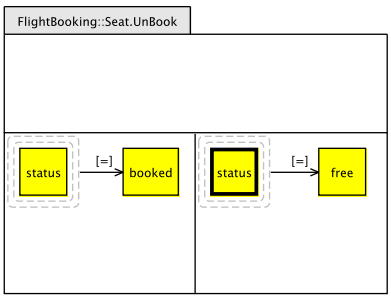


(b) AD of operation `GetFreeSeatsForFlightOfClass`

Figure D.27: VCL CDs and ADs of the behavioural model of flight booking

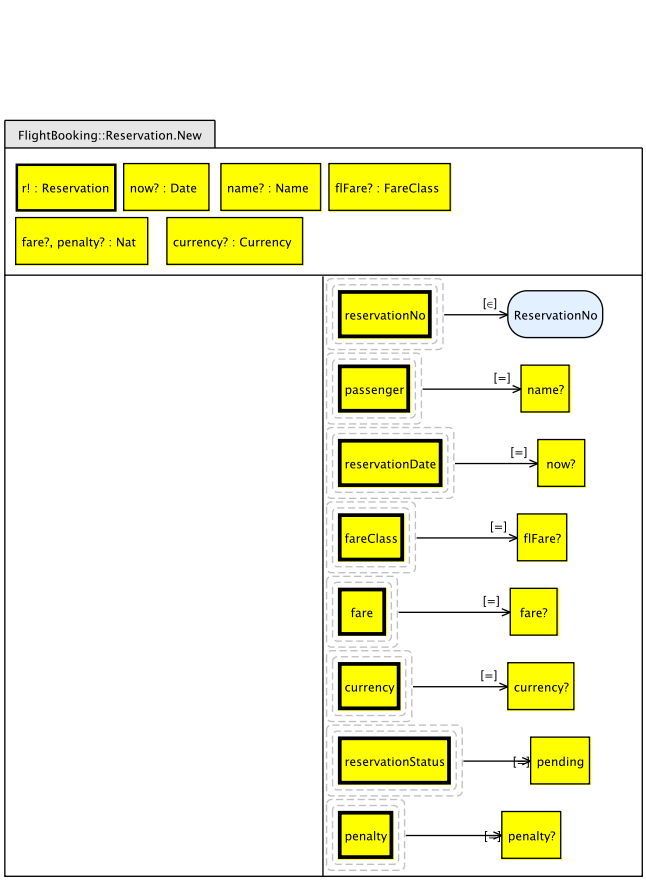


(a) CD of operation `Seat.Book`

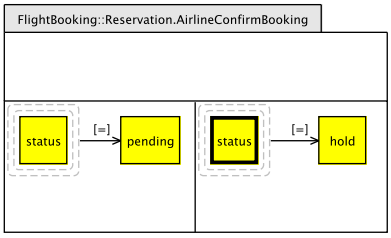


(b) CD of operation `Seat.UnBook`

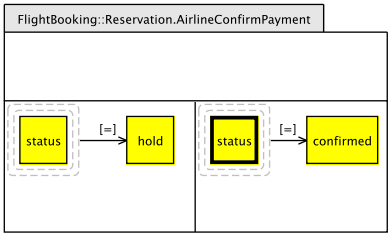
Figure D.28: VCL CDs and ADs of the behavioural model of flight booking



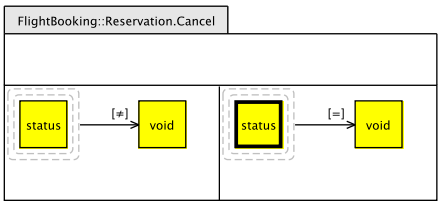
(a) CD of operation Reservation.New



(b) CD of operation Reservation.AirlineConfirmBooking

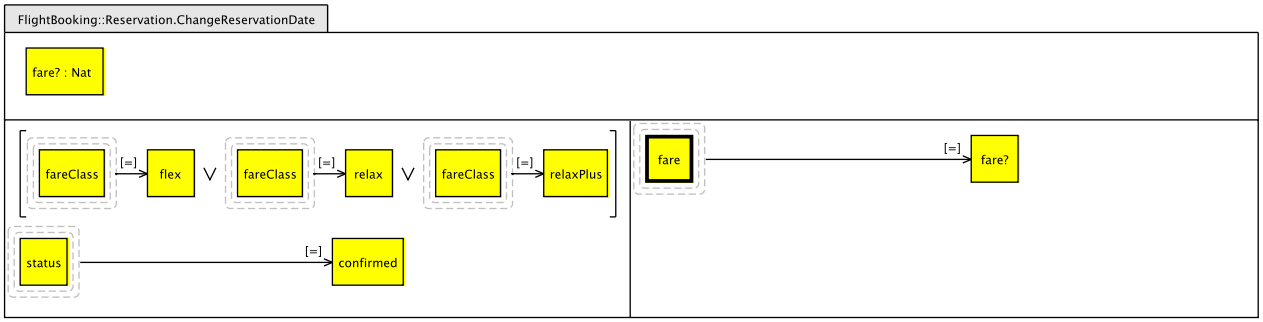


(c) CD of operation Reservation.AirlineConfirmPayment

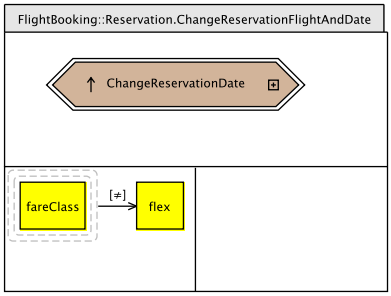


(d) CD of operation Reservation.Cancel

Figure D.29: VCL CDs and ADs of the behavioural model of flight booking



(a) CD of operation Reservation.ChangeReservationDate



(b) CD of operation Reservation.ChangeReservationFlightAndDate

Figure D.30: VCL CDs and ADs of the behavioural model of flight booking

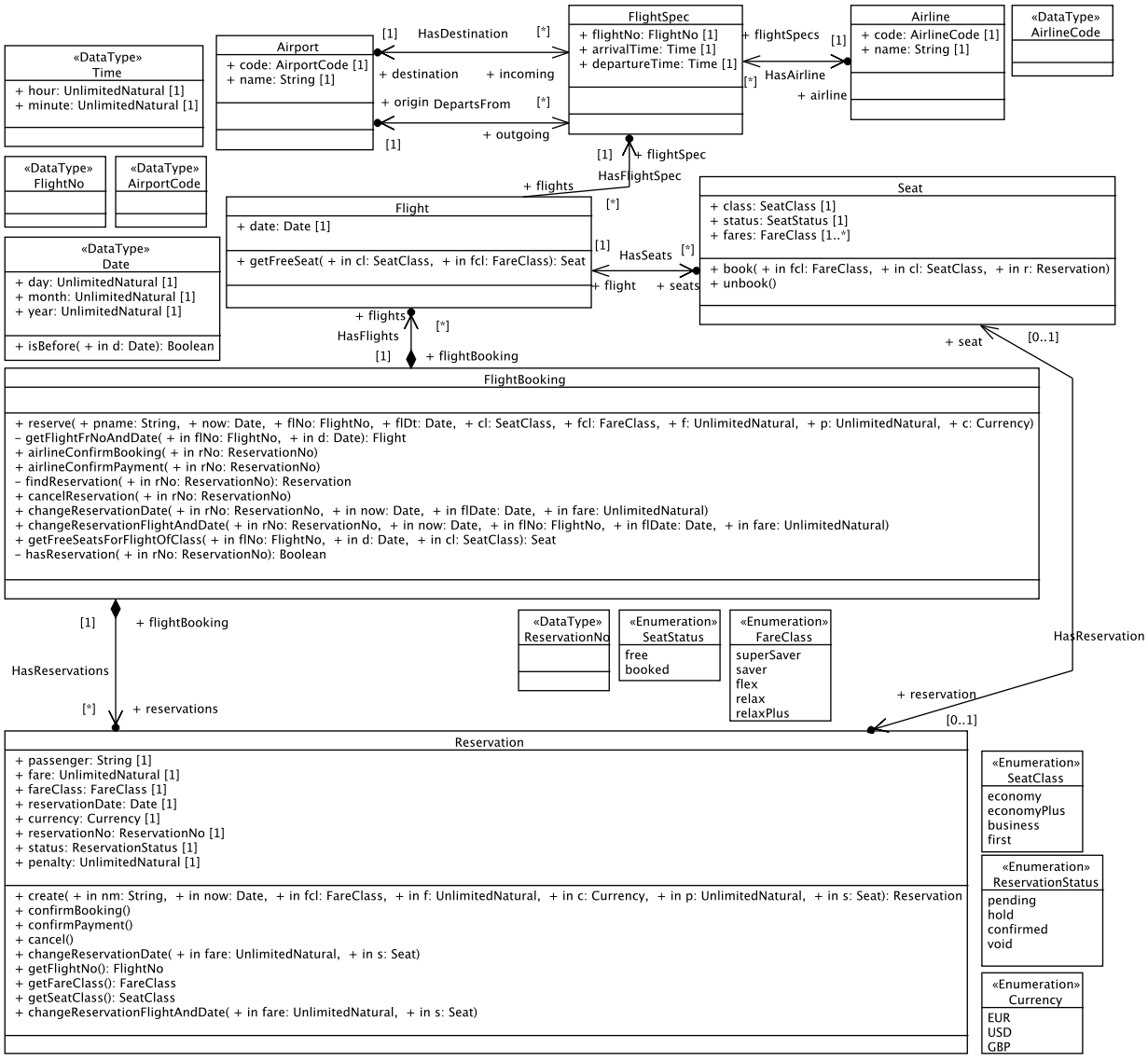


Figure D.31: UML Class diagram of flight booking

```
context FlightBooking getFlightFrNoAndDate(flNo : FlightNo, d : Date) : Flight
post: result = self.flights->select(f | f.flightSpec.flightNo = flNo and f.date = d)->any(true)
```

(a) OCL operation *FlightBooking* :: *getFlightFrNoAndDate*()

```
context FlightBooking reserve(pname : String, now : Date, flNo : FlightNo, flDt : Date,
  cl : SeatClass, fcl : FareClass, f : UnlimitedNatural, p : UnlimitedNatural, c : Currency)
pre: now.isBefore(flDt)
post:
  let f = getFlightFrNoAndDate(flNo, flDt),
      s = f.getFreeSeat(cl, fcl),
      r = Reservation::create(pname, now, fcl, f, c, p, s)
  in
    s.book(fcl, cl, r) and self.reservations = self.reservations@pre->union(Set{r})
```

(b) OCL operation *FlightBooking* :: *reserve*()

```
context FlightBooking hasReservation(rNo : ReservationNo) : Boolean
post: result = reservations->exists(r | r.reservationNo = rNo)
```

(c) OCL operation *FlightBooking* :: *hasReservation*()

```
context FlightBooking findReservation(rNo : ReservationNo) : Reservation
pre: hasReservation(rNo)
post: result = reservations->any(reservationNo = rNo)
```

(d) OCL operation *FlightBooking* :: *findReservation*()

```
context FlightBooking airlineConfirmBooking(rNo : ReservationNo)
pre: hasReservation(rNo)
post: let r = findReservation (rNo)
  in r.confirmBooking()
```

(e) OCL operation *FlightBooking* :: *airlineConfirmBooking*()

```
context FlightBooking airlineConfirmPayment(rNo : ReservationNo)
pre: hasReservation(rNo)
post: let r = findReservation(rNo)
  in r.confirmPayment()
```

(f) OCL operation *FlightBooking* :: *airlineConfirmPayment*()

```
context FlightBooking cancelReservation(rNo : ReservationNo)
pre: hasReservation(rNo)
post: let r = findReservation(rNo)
  in r.cancel()
```

(g) OCL operation *FlightBooking* :: *cancelReservation*()

```
context FlightBooking changeReservationDate(rNo : ReservationNo, now : Date,
  flDate : Date, fare: UnlimitedNatural)
pre: hasReservation(rNo) and now.isBefore(flDate)
post:
  let r = findReservation(rNo),
     f = getFlightFrNoAndDate (r.getFlightNo(), flDate),
     s = f.getFreeSeat(r.getClassSeat(), r.getFareClass())
  in r.changeReservationDate(fare, s)
```

(h) OCL operation *FlightBooking* :: *changeReservationDate*()

Figure D.32: OCL operations of flight booking

```
context FlightBooking changeReservationFlightAndDate(rNo : ReservationNo, now : Date,
  flDate : Date, fare: UnlimitedNatural)
pre: hasReservation(rNo) and now.isBefore(flDate)
post:
  let r = findReservation(rNo),
      f = getFlightFrNoAndDate (r.getFlightNo(), flDate),
      s = f.getFreeSeat(r.getSeatClass(), r.getFareClass())
  in r.changeReservationFlightAndDate(fare, s)
```

(a) OCL operation *FlightBooking* :: *changeReservationFlightAndDate*()

```
context FlightBooking getFreeSeatsForFlightOfClass
  (flNo : FlightNo, d : Date, cl : SeatClass) : Seat
post: let f = getFlightFrNoAndDate(flNo, d)
  in result = f.seats->select(s | s.status = SeatStatus::free and s.class = cl)
```

(b) OCL operation *FlightBooking* :: *getFreeSeatsForFlightOfClass*()

```
context FlightBooking getFreeSeat(cl : SeatClass, fcl : FareClass) : Set Seat
post: result = self.seats->select (s | s.class = cl and s.fares->includes(fcl)
    and s.status = SeatStatus::free)->any(true)
```

(c) OCL operation *Flight* :: *getFreeSeat*()

```
context Seat book(fcl : FareClass, cl : SeatClass, r : Reservation)
pre: status = SeatStatus::free and class = cl and fares->includes(fcl)
post: status = SeatStatus::booked and reservation = r
```

(d) OCL operation *Seat* :: *book*()

```
context Seat unbook(fcl : FareClass, cl : SeatClass, r : Reservation)
pre: status = SeatStatus::booked
post: status=SeatStatus::free and reservation->isEmpty()
```

(e) OCL operation *Seat* :: *unbook*()

Figure D.33: OCL operations of flight booking

```
context Reservation
  create(nm : String, now: Date, fcl : FareClass, f : UnlimtedNatural, c : Currency,
    p : UnlimtedNatural, s : Seat) : Reservation
post: self.reservationNo = ReservationNo.allInstances->any(true)
  and self.passenger = nm and self.fareClass = fcl
  and self.reservationDate = now and self.currency = c
  and self.fare = f and self.penalty = p and seat = s
  and status = ReservationStatus::pending and result = self
```

(a) OCL operation *Reservation* :: *create*()

```
context Reservation confirmBooking ()
pre: self.status = ReservationStatus::pending
post: self.status = ReservationStatus::hold
```

(b) OCL operation *Reservation* :: *confirmBooking*()

```
context Reservation confirmPayment ()
pre: self.status = ReservationStatus::hold
post: self.status = ReservationStatus::confirmed
```

(c) OCL operation *Reservation* :: *confirmPayment*()

```
context Reservation cancel ()
pre: self.status <> ReservationStatus::void
post: seat.unbook() and status = ReservationStatus::void
```

(d) OCL operation *Reservation* :: *cancel*()

```
context Reservation changeReservationDate (fare : UnlimitedNatural, s : Seat)
pre: (fareClass  = FareClass::flex or fareClass  = FareClass::relax
    or fareClass  = FareClass::relaxPlus)
  and status = ReservationStatus::confirmed
post: self.fare = fare and seat@pre.unbook() and seat = s
```

(e) OCL operation *Reservation* :: *changeReservationDate*()

```
context Reservation changeReservationFlightAndDate (fare : UnlimitedNatural, s : Seat)
pre: status <> ReservationStatus::flex
post: changeReservationDate(fare, s)
```

(f) OCL operation *Reservation* :: *changeReservationFlightAndDate*()

```
context Reservation getFlightNo () : FlightNo
post: result = self.seat.flight.flightSpec.flightNo
```

(g) OCL operation *Reservation* :: *getFlightNo*()

```
context Reservation getFareClass () : FareClass
post: result = fareClass
```

(h) OCL operation *Reservation* :: *getFareClass*()

```
context Reservation getSeatClass () : SeatClass
post: result = self.seat.class
```

(i) OCL operation *Reservation* :: *getSeatClass*()

Figure D.34: OCL operations of flight booking

Appendix E

Questionnaires

This appendix presents the questionnaires that have been designed to support the experiment presented here. They are as follows:

- The problem comprehension questionnaires (section E.2) try to evaluate how well the subjects comprehended the system (or problem) that they modelled.
- The model comprehension questionnaires (section E.4) try to evaluate how well the subjects comprehended the given solution models of the experiment case studies.
- The debriefing questionnaire (section E.5) gathers the impressions of subjects concerning the languages under study following their experiment experience.

E.1 Ability Assessment

Edit this form

Participant Questionnaire for the VCL vs UML+OCL Experiment

This is the questionnaire for the participants of a software design modelling experiment. This experiment will compare the Visual Contract Language (VCL) against UML and OCL.

Please complete this form. It will enable us to categorise you as user, which will help us assessing the amount of training that you need and to obtain accurate results from our study.

The contents of this form are absolutely confidential. They are to be used for scientific purposes only. Information identifying the respondent will not be disclosed under any circumstances.

Thank you for your cooperation!

*Required

Your Name: *

Your Email: *

What is your gender? *

- Male
- Female

What is your occupation? *

- Bachelor's student
- Master's student
- PhD Student
- Researcher
- Software Developer
- Other:

What is the highest level of formal education that you have completed? *

- High-school
- Bachelor

- ○ Master
- ○ Doctorate

Have you completed or are in the process of completing a formal degree in Computer Science? *

- ○ Yes
- ○ No

If you have completed a formal degree in computer science, what degree levels have you completed?

- ☐ Bachelor's
- ☐ Master's
- ☐ PhD

Please provide further comments on your computer science knowledge and education. If you are currently undertaking a computer science degree, please state the year of your studies.

How many assessed computer programming courses did you complete so far? *

- ○ None
- ○ 1
- ○ 2
- ○ 3+

How much computer programming experience do you have in producing "real-world" software that actually has to be used in some context? *

- ○ None
- ○ <1
- ○ 1-2 years
- ○ 3+ years

Please select the computer programming subjects that you have covered as part of your computer science education.

- ☐ Algorithms and data structures
- ☐ Functional Programming

- ☐ Imperative Programming
- ☐ Logic Programing

Please indicate other computer programming subjects that you have completed and are relevant to your computer programming knowledge, if you feel necessary

Thinking of your experience in ``real-world'' software development projects that required at least 3 months active programming effort from your part. In how many such projects have you participated? *

- ○ None
- ○ 1
- ○ 2
- ○ 3+

With respect to your proficiency with different programming languages, please indicate the extent to which you agree or disagree with the following statements. Use the rows named "other" for programming languages that you have used before and that are not listed. *

Levels of agreement: 1: Strongly agree; 2: Agree; 3: Uncertain; 4: Disagree; 5: Strongly disagree

	1	2	3	4	5
Java	○	○	○	○	○
C/C++	○	○	○	○	○
C#	○	○	○	○	○
Other A	○	○	○	○	○
Other B	○	○	○	○	○
Other C	○	○	○	○	○

Please write down the languages that you have specified as "other" in the previous question, if you have some proficiency in them.

Based on your programming experience and performance on programming courses, please indicate the extent to which you agree or disagree with the following statement: I am a proficient computer programmer. *

	1	2	3	4	5	
Strongly Agree	○	○	○	○	○	Strongly Disagree

How many assessed courses have you done that involved the subjects of discrete mathematics, logic and set theory? *

- ○ None
- ○ 1
- ○ 2
- ○ 3+

Based on your experience in using discrete mathematics, logic and set theory, and your performance in courses on these subjects, please indicate the extent to which you agree or disagree with the following statements. *

Levels of agreement: 1: Strongly agree; 2: Agree; 3: Uncertain; 4: Disagree; 5: Strongly disagree

	1	2	3	4	5
In general, I am well-prepared	○	○	○	○	○
I can master the concepts of set theory	○	○	○	○	○
I can master the concepts of predicate logic	○	○	○	○	○
I can master advanced concepts, such as proof	○	○	○	○	○

How many assessed courses have you done on object-oriented design modelling, involving the UML or other object-oriented diagrammatic notation, such as OMT or Booch? *

- ○ None

- 1
- 2
- 3+

How many of these courses required an assessed (or graded) project where you had to build a design model using UML or other diagrammatic notation? *

- None
- 1
- 2
- 3+

Did any of these courses on software design modelling involved the Object Constraint Language (OCL)? *

- Yes, OCL was briefly discussed
- Yes, OCL was extensively covered
- No

Please indicate the names of the object-oriented design courses that you have completed.

How many of these courses with an assessed (or graded) modelling project also required modelling using OCL? *

- None
- 1
- 2
- 3+

Thinking of your experience in software design modelling projects using diagrammatic notations that required at least 3 months active modelling effort from your part. In how many such projects have you participated? *

- None
- 1
- 2
- 3+

Thinking of your experience in software design modelling projects that involved OCL

and that required at least 3 months active modelling effort from your part. In how many such projects have you participated? *

- None
- 1
- 2
- 3+

Based on your experience in software design modelling using diagrammatic notations such as UML and your performance in related courses, please indicate the extent to which you agree or disagree with the following statements. *

Levels of agreement: 1: Strongly agree; 2: Agree; 3: Uncertain; 4: Disagree; 5: Strongly disagree

	1	2	3	4	5
In general, I am well prepared in object-oriented design	○	○	○	○	○
I can master advanced concepts, such as behavioural modelling	○	○	○	○	○
I am a proficient UML modeller	○	○	○	○	○
I am proficient with OCL	○	○	○	○	○
I am a proficient modeller using some other diagrammatic notation	○	○	○	○	○

Please specify the "other diagrammatic notation" of the previous question if you have some proficiency in it.

Please provide further comments on your object-oriented design modelling knowledge and experience if you feel necessary.

How many assessed courses have you done that involved software design modelling

using a notation with a strong mathematical flavour, such as Z, B or Alloy? *

- ○ None
- ○ 1
- ○ 2
- ○ 3+

How many of these courses required an assessed (or marked) project that actually involved building a formal model? *

- ○ None
- ○ 1
- ○ 2
- ○ 3+

Please state the designations of the formal modelling courses that you have completed.

What mathematical-based notations of software design have you used?

Tick all that apply

- ☐ Alloy
- ☐ Z/Object-Z
- ☐ B/Event-B
- ☐ VDM
- ☐ Other:

Thinking of your experience in software design modelling projects using formal notations, such as Z, B or Alloy, that required at least 3 months active modelling effort from your part. In how many such projects have you participated? *

- ○ None
- ○ 1
- ○ 2
- ○ 3+

Based on your experience in software design modelling using formal notations, such as Z, B or Alloy, and your performance in related courses, please indicate the extent to which you agree or disagree with the following statement. *

Levels of agreement: 1: Strongly agree; 2: Agree; 3: Uncertain; 4: Disagree; 5: Strongly disagree

	1	2	3	4	5
I feel proficient with at least one formal modelling notation	○	○	○	○	○
I feel proficient using at least one theorem prover	○	○	○	○	○
I feel proficient using at least one model-checker or sat solver	○	○	○	○	○

Please provide further comments on your knowledge and experience of formal modelling languages, such as Z, B or Alloy, if you feel necessary.

Have you received any prior training in the Visual Contract Language (VCL)?

- ○ Yes
- ○ No

Have you ever used VCL in a project? *

- ○ Yes
- ○ No

If you answered "Yes" to the previous question, please specify the context in which you used VCL

Submit

Never submit passwords through Google Forms.

E.2 Problem Comprehension, University Library

E.2.1 University Library

Edit this form

VCL Problem Comprehension Task: University Library.

This is a comprehension questionnaire for the experiment VCL vs UML+OCL. It tries to assess the understanding of the University Library case study that you've gained while modelling this case study in VCL. You have 15 minutes to complete this task.

*Required

Your Name: *

Please state whether the following descriptions of states are either possible or impossible in the university library system.

	Possible	Impossible
A restricted copy of "Design Patterns" book being borrowed by taught student Adam Perry	○	○
A one-week copy with status 'shelved' being borrowed by taught student Alexander Gagarin	○	○
The 1975 and 1995 editions of title "The Mythical Man-Month" issued as separate library books with different publication years and ISBNs where each book has a one-month copy each	○	○
The books "Design Patterns" and "Applying UML and Patterns" with the book number 78654	○	○
A recalled copy of the book "Distributed systems: Principles and Paradigms" being borrowed by student Véronique Streff	○	○
Research student "Marie Piaf" borrowing all 10 library copies of book "UML Distilled"	○	○
Taught students "João Mateus" and "Maria Pacheco" with library number "01235" borrowing 10 books each	○	○

Please state whether the following operations of the university library system are either valid or invalid.

	Valid	Invalid
A research student with 15 borrowings returns a copy of the book "Design Patterns" and borrows a copy of the book "Java in a Nutshell" immediately afterwards	○	○
A staff member with 25 borrowings borrows a one-month copy of the book "Applying UML and Patterns".	○	○
A taught student with 3 borrowings borrows a restricted copy of the book "Principia Mathematica"	○	○
At the library desk, a taught student with 10 borrowings does a successful renewing of one borrowing and borrows a copy of the "Compilers" book	○	○

The library is considering allowing staff members to borrow 'restricted' book copies. How could this situation be accommodated in a model of the current requirements description?

○ No need to model it. The current model already allows this situation.

○ Remove all existing invariants concerning 'restricted' copies'. Put an invariant that says that restricted copies can only be borrowed by staff members.

○ Remove all existing invariants concerning 'restricted' copies'. Put one pre-condition on borrows allowing borrowings provided the member doing the borrowing is not staff and the copy is not restricted.

○ No need to change invariants. All is needed is an extra pre-condition on the operation borrows allowing the operation provided the copy is not restricted or the member is staff

Which of the following circumstances describes a valid 'borrow' operation in the University Library system?

○ A taught student with 10 borrowings borrows a one-week copy of the book "The Mythical man-month"

○ A research student with 14 borrowings borrows a one-month copy of a book that has a 'one-week' copy, which he is already borrowing.

○ A staff member with 25 borrowings returns to the library a copy that has been recalled, which he borrows immediately afterwards

○ A taught student with 9 borrowings borrows a copy that she is already borrowing

Which of the following describes a valid operation of the university library system?

○ A taught student with 5 borrowings renews a borrowed copy that the student has already renewed 3 times

- A staff member with 7 borrowings recalls a restricted copy of the book "Java for Dummies"
- A research student with 10 borrowings returns a copy that has been recalled
- A research student with 14 borrowings collects a previously recalled copy whose status is "shelved"

Which of the following describes an invalid operation of the university library system?

- A taught student with 10 borrowings returns a book that is due on the 25/03/2013 on the 26/03/2013
- On the 26/03/2013, a staff member borrows a one-month copy that is recalled and as a result the new due date becomes 26/04/2013
- A taught student with 10 borrowings renews a one-week copy on the 23/03/2013 and the due date becomes 30/03/2013 for a copy that is not recalled and has not been renewed before
- A research student with 15 borrowings returns a recalled copy due on the 13/03/2013 on the 17/03/2013

Submit

Never submit passwords through Google Forms.

Edit this form

UML+OCL Problem Comprehension Task: University Library

This comprehension questionnaire of the experiment VCL vs UML+OCL tries to assess the comprehension of the University Library case study that you've gained through modelling in UML and OCL. You have 15 minutes to complete this task.

*Required

Your Name: *

Please state whether the following descriptions of states are either possible or impossible in the University Library system.

	Possible	Impossible
A copy of 1st edition of book "The Mythical Man-Month" and the same copy associated with another book with the same title, but a different year and ISBN.	○	○
Book entitled "Principia Mathematica" with authors "Bertrand Russel" and "Alfred Whitehead"	○	○
Book "Design Patterns" with one restricted copy with status 'toCollect' and one one-month copy with status 'recalled'.	○	○
Taught students John Smith and António Santos holding recalls for the same copy of book "Using Z".	○	○
Research students Giovanni Moretti and Marie Aubry borrowing the same copy of book "A discipline of programming."	○	○
Member James Oliver successfully borrows a one-week copy of "Real World Research", although he did not collect a previously recalled copy (now with status 'toCollect') from the library	○	○

	Possible	Impossible
Taught student "Cristiano Messi" with more borrowings than staff Member "Anthony Hoare"	○	○

The university library is considering a fourth category of 'Member' called 'alumni' for former students. 'Alumni' members can borrow at most 5 books of category one-month only. Please state whether the following statements concerning possible modelling solutions for this requirement are either valid or invalid.

	Valid	Invalid
The structural model would base the modelling of this solution on the existing 'alumni' member category	○	○
The existing invariant(s) on borrowing entitlements would be extended to say that alumni can borrow at most 5 copies	○	○
An extra invariant would say that the borrowed copies of 'alumni' members must be one-week	○	○
A pre-condition on the operation borrows would check that the member is either not 'alumni' or that the copy is 'one-month'	○	○

Consider the following situation. A research student is borrowing a one-month copy of a book that has been recalled and is due tomorrow. She doesn't need the book right now, but she is definitely going to need it in one and a half weeks time. What would you advise her to do in order to get a copy of the desired book when she absolutely needs it, but remember that she is on a tight budget?

○ It's impossible to get the library copy of the book when she needs it. Better buy one from Amazon.

○ Return the book copy and wait for the book copy to be returned.

○ Keep the book and pay the costly library fines associated with not returning recalled books

○ Return the book as soon as possible and recall the book again

Which of the following circumstances describes a valid recall operation in the university library system?

○ A research student with 15 borrowings recalls a copy of a book he is already borrowing

○ A staff member with 25 borrowings recalls a one-month copy of the book "Applying UML and Patterns" when there are no other copies of this book available for borrowing in the library, but one of these copies is due the day after

○ A research student with 3 borrowings recalls a restricted copy of the book "Principia

Mathematica"

- A taught student with 10 borrowings recalls a 'Copy' whose status is 'toCollect'

Which of the following operations of the university library system is invalid?

- A taught student with 9 borrowings renews a book on the 25/04/2013 that is due on the 24/04/2013
- On the 21/04/2013, a research student with 15 borrowings returns a book that is due on the 25/04/2013
- On the 26/05/2013 , a research student with 1 borrowing returns a one-week copy of the book "Latex Companion" that is due on 25/04/2013
- A staff member with 25 borrowings renews on the 02/05/2013 a borrowing that he has not renewed before, it is not recalled and it's due on the 02/05/2013

Which of the following operations of the university library system is valid?

- A staff member recalls a restricted copy of "The B Book" being borrowed by a taught student
- A research student with 10 borrowings collects a previously recalled copy of book "Software Engineering" whose status is 'recalled'
- A taught student with 10 borrowings returns on the 5/5/2013 a recalled borrowing of book "UsingUML" that is due on 2/5/2013
- A staff member recalls a copy of book "UML Distilled" that he is currently borrowing

Submit

Never submit passwords through Google Forms.

E.2.2 Flight Booking

Edit this form

VCL Problem Comprehension Task: Flight Booking

This is a comprehension questionnaire for the experiment VCL vs UML+OCL. It tries to assess your understanding of the flight booking case study that you've gained while performing the modelling task. You have 15 minutes to complete this task.

*Required

Your Name: *

Please state whether the following descriptions of states should either be possible or impossible in the flight booking system.

	Possible	Impossible
Carrier 'TAP' with flights 'TP2345' and 'TP2346' departing from 'Lisbon' to 'London Heathrow' on the same day	○	○
A passenger with two different reservations for two seats on the same flight	○	○
Flight 'BA3567' going from 'London Heathrow' to 'Sydney' and then 'Perth'	○	○
A passenger holding a reservation with status 'hold' and fare class 'flex' for an 'economy' seat in flight 'AF3453' from Paris to Martinique	○	○
A passenger holding reservation 'R345678' with status 'confirmed' and fare class 'saver' for a 'economyPlus' seat with status 'booked' in flight 'KLM2384' from 'Amsterdam' to 'New York'	○	○
A passenger holding a 'flex' reservation with status 'confirmed' to a seat with status 'booked' and class 'business'	○	○

	Possible	Impossible
A passenger with a 'relaxPlus' reservation with status 'confirmed' that is not associated with a flight seat	○	○

The flight booking system is considering not guaranteeing a flight seat for reservations made with the 'superSaver' fare class: if the flight is fully booked, such seats may be taken. Please state whether the following statements concerning changes to the flight booking model to accommodate this new requirement are either valid or invalid.

	Valid	Invalid
The structural model would be extended to include a 'superSaver' fare class	○	○
There would be an invariant saying that seats with 'superSaver' reservations may be taken	○	○
There would be a change to the 'Reserve' operation to obtain seats that have been reserved with the 'superSaver' fare if there are no more free seats	○	○
The operations 'changeReservationDate' and 'changeReservationFlightAndDate' would be changed to disallow such changes for the 'superSaver' fare class	○	○

Which of the following circumstances describes a valid 'Change Flight and Date of Reservation' operation in the system?

○ On the 15/5/2013, a passenger holding a reservation with status 'confirmed' and fare class 'relax' exchanges an economy seat in flight 'KL6061' on the 05/06/2013 for an economy seat in flight 'LH4561' on the '06/06/2013'

○ On the 17/4/2013, a passenger holding a reservation with status 'hold' and fare class 'relaxPlus' exchanges a 'first' seat in flight 'BA8143' on the 06/06/2013 for an 'economy' seat in flight 'BMI6721' on the '06/07/2013'

○ On the 8/5/2013, a passenger holding a reservation with status 'confirmed' and fare class 'flex' exchanges a 'business' seat of flight 'TR6754' on the 06/05/2013 for a business seat for flight 'TR5421' on the '06/06/2013'

○ On the 4/4/2013, a passenger holding a reservation with status 'confirmed' and fare class 'relax' exchanges a 'business' seat of flight 'AF2384' on the 03/04/2013 for an economy seat on flight 'DL6721" on the '06/06/2013'

A passenger has a reservation with status 'confirmed' for a first seat with a 'relaxPlus' fare for the flight 'TR1494' from Istanbul to Rome on the 25/05/2013. Unfortunately, due to unforeseen circumstances, he can no longer take the reserved flight. He wants to fly the next day, but the reservation's carrier, Turkish airlines, arrives later than he would wish for.

There is an Alitalia flight to Rome from Istanbul earlier. His priorities are, in descending order: (a) spend the least amount of money, (b) keep his 'first' class seat as he loves luxury and (c) don't get to Rome too late. Which of the following describes a valid and accurate situation that takes this passenger's priorities into account?

- Change the reservation to the Alitalia flight using the operation 'Change Flight and Date of Reservation' , which means he doesn't get to Rome too late and he keeps his first seat.
- Change the reservation's date to the Turkish airlines flight using the operation 'Change Date of Reservation' , which means he doesn't lose too much money (just the reservation's penalty) and he keeps his first seat.
- Change the reservation's date to the Turkish airlines flight using the operation 'Change Date of Reservation' , which means he doesn't lose his first seat, he doesn't spend money and he gets there at a very convenient time.
- Cancel the reservation, get a refund and buy a first ticket with Alitalia. This means he can get to Rome at a convenient time, even if he spends a fair amount of money.

Which of the following system operations is valid?

- On the 02/03/2013, a passenger reserves, with fare class 'flex', an 'economy' seat with status 'free' in flight 'LG 3838' from Luxembourg to Geneva for the 27/02/2013
- On the 01/03/2013, a passenger reserves a 'free' 'economyPlus' seat with fare class 'saver' in flight 'LH7813' from 'Frankfurt' to 'Singapore' of 1/5/2013
- On the 02/04/2013, a passenger with a 'flex' fare reservation for an 'economy' seat in flight 'AF 3043' on the 4/5/2013 from 'Paris' to 'Milan' changes the reservation to an 'economy' seat in the same flight for the 5/5/2013
- On the 05/06/2013, a passenger reserves, with a 'flex' fare, a 'first' seat with status 'free' in flight 'BA2394' from 'London Heathrow' to 'Los Angeles' on the 06/06/2013

Which of the following operations is invalid?

- The airline confirms the flight payment for the reservation of passenger 'Bullent Akin' on flight 'TR4388' from Ankara to Budapest; the reservation's status changes to 'confirmed'
- On the 05/06/2013, a passenger changes the date of his reservation with status 'confirmed' and fare class 'flex' for an 'economy' seat in flight 'KL6061' from Amsterdam to San Francisco from the 04/06/2013 to the 07/07/2013
- On 4/5/2013, a passenger changes the date of a 'relaxPlus' 'first' reservation with status 'confirmed' for an 'IB4567' flight from Barcelona to Rio de Janeiro from the 6/6/2013 to the 7/7/2013
- The airline confirms the reservation of passenger 'José Paquito' on flight 'IB3278' from Madrid to Barcelona; the reservation's status changes to 'hold'.

Submit

Never submit passwords through Google Forms.

Edit this form

UML+OCL Problem Comprehension Task: Flight Booking

This comprehension questionnaire of the experiment VCL vs UML+OCL tries to assess your understanding of the flight booking case study after the modelling tasks. You have 15 minutes to complete this task.

*Required

Your Name: *

Please state whether the following descriptions of states should either be possible or impossible in the flight booking system.

	Possible	Impossible
A passenger with reservation 'RLG8789' whose status is 'void'	○	○
Carrier 'LuxAir' with flight number 'LG3537' departing from 'Luxembourg' and with destinations 'Porto' and 'London	○	○
A passenger with reservation 'RTP6384' holding seats for flights 'TP345' from 'Recife' to 'Lisbon' and 'TP686' from 'Lisbon' to 'Luxembourg'	○	○
A passenger holding reservation 'RBA7654' with status 'hold' and fare class 'saver' for an 'economy' seat with status 'booked' in flight 'BA8143' from 'London Gatwick' to 'Bridgetown'	○	○
A passenger holding reservation 'RFA7895' with status 'confirmed' and fare class 'saver' for a 'business' seat with status 'booked' in flight 'AF2384' from 'Paris CDG' to 'Tunis'	○	○
Carrier 'KLM' with two flights numbered 'KL6061' from 'Amsterdam' to 'Portland' departing at different times on	○	○

	Possible	Impossible
the 06/05/2013		
A passenger holding a reservation with status 'pending' for an economy seat with 'supersaver' fare and status 'booked' for flight 'LH 1272' from Frankfurt to Innsbruck	○	○

The flight booking system is considering a new 'comfort' class fare, which should apply to 'first' seats only and should provide the same entitlements as fares 'relax' and 'relaxPlus' concerning alterations to reservations. Please state whether the following statements concerning changes to a flight booking model of the original requirements to accommodate this new requirement are either valid or invalid.

	Valid	Invalid
The structural model would need to be extended to include a new 'comfort' class fare	○	○
The existing invariants concerning fare class restrictions would be extended to say that the 'comfort' class fare applies to 'first' and 'business' seats only	○	○
The pre-condition of operation 'ChangeReservationDate' would need to allow this operation for the 'comfort' fare class.	○	○
The pre-condition of operation 'ChangeFlightAndDateOfReservation' would need to be extended to allow this operation for the 'comfort' fare class when the reserved seat is not of class 'first'.	○	○

Which of the following circumstances describes a valid 'Change Reservation Date' operation in the system?

○ On the 4/4/2013, a passenger holding a reservation with status 'confirmed' and fare class 'saver' changes an 'economy' seat for flight 'TP6568' for the 06/05/2013 to the '06/06/2013'

○ On the 15/3/2013, a passenger holding a reservation with status 'hold' and fare class 'flex' changes an 'economy' seat for flight 'BA8143' on the 06/06/2013 to the '06/07/2013'

○ On the 5/3/2013, a passenger holding a reservation with status 'confirmed' and fare class 'relax' changes an 'business' seat for flight 'AF2384' on the 02/03/2013 to the '06/06/2013'

○ On the 7/4/2013, a passenger holding a reservation with status 'confirmed' and fare class 'relaxPlus' changes a 'first' seat in flight 'KL6061' on the 05/06/2013 to the '06/06/2013'

Which of the following circumstances describes a valid 'Change Reservation Date' operation in the system?

- On the 4/4/2013, a passenger holding a reservation with status 'confirmed' and fare class 'saver' changes an 'economy' seat in flight 'TP6568' for the 06/05/2013 to the '06/06/2013'
- On the 15/3/2013, a passenger holding a reservation with status 'hold' and fare class 'flex' changes an 'economy' seat in flight 'BA8143' on the 06/06/2013 to the '06/07/2013'
- On the 5/3/2013, a passenger holding a reservation with status 'confirmed' and fare class 'relax' changes a 'business' seat in flight 'AF2384' on the 02/03/2013 to the '06/06/2013'
- On the 7/4/2013, a passenger holding a reservation with status 'confirmed' and fare class 'relaxPlus' changes a 'first' seat in flight 'KL6061' on the 05/06/2013 to the '06/06/2013'

A passenger holds a reservation with status 'confirmed' for an economy seat with a 'flex' fare for the flight 'LX1494' from Zurich to Prague on the 27/06/2013. Unfortunately, due to unforeseen circumstances, he can no longer fly on the 27/06/2013 and he must absolutely fly the day after, but the reservation's carrier ('LX', Swiss Airways) does not have a flight to Prague on the 28th. There is, however, flight 'OK 5593', of Czech Airlines, available on the 28th. Which of the following describes a valid and accurate way to reserve a seat for flight 'OK5593'?

- Change the reservation for flight 'OK5593' using the operation 'Change Date of Reservation'
- Change the reservation for flight 'OK5593' using the operation 'Change Flight and Date of Reservation'
- Cancel the current reservation and make another one, which does not incur in a loss of money as the reservation for flight 'LX1494' has not been paid
- Cancel the current reservation, get the refund if any, make another reservation for flight 'OK5593' and pay for it

Which of the following system operations is valid?

- On the 01/02/2013, a passenger reserves with fare class 'flex' an 'economy' seat with status 'booked' on flight 'LG 3837' from Luxembourg to Madrid on the 25/03/2013
- On the 04/03/2013, a passenger reserves a 'free' 'business' seat with fare class 'saver' on flight 'BA7812' from 'London' to 'New York' of 1/5/2013
- On the 02/04/2013, a passenger reserves, with a 'flex' fare, an 'economy' seat with status 'free' on flight 'AF2384' from 'Paris' to 'Tunis' on the 03/04/2013.
- A passenger holding a 'flex' fare reservation for an 'economy' seat on flight 'BA 8143' from 'Luxembourg' to 'London Gatwick' changes the reservation to an 'economy' seat on flight 'LG 4595' from Luxembourg to 'London City'

Which of the following operations is invalid?

- A passenger cancels a 'relaxPlus' reservation with status 'pending' for an 'economy' seat on flight 'IB4567' from Madrid to Miami
- The airline confirms the payment of passenger David Stewart's reservation on flight 'BA2586' from London Gatwick to Venice; the reservation status becomes 'confirmed'
- The airline confirms the booking of passenger 'José Ferreira' on flight 'TP1621' from Lisbon to Funchal; the reservation status becomes 'pending'
- On the 03/02/2013, a passenger changes the date of his reservation with status 'confirmed' and fare class 'flex' for flight 'KL6061' from Amsterdam to Portland from the 05/05/2013 to the 15/05/2013

Submit

Never submit passwords through Google Forms.

E.3 Defect Detection, University Library

E.3.1 University Library

Edit this form

University Library. VCL Defect Detection Task

You were given a VCL model with seeded defects and your task is to identify them. You have 25 minutes to complete this task.

*Required

Your Name: *

Please identify the defects in the given VCL structural diagram.

Please identify the defects in the invariants of the given VCL model.

Go through the AD corresponding to each assertion that is included in the VCL strcutural diagram.

Please identify the defects in the VCL behaviour diagram

Please identify the defects in the operations of the given VCL model.

Go through each contrat and assertion diagram that is included in the behaviour diagram. Don't forget to check all included assertions and contracts in the contract and assertion diagrams as

well.

Submit

Never submit passwords through Google Forms.

Powered by

This content is neither created nor endorsed by Google.

Report Abuse - Terms of Service - Additional Terms

Edit this form

UML+OCL. Uni Library. Defect Detection Task

You were given a UML+OCL model with seeded defects. Your task is to identify the defects in the model. You have 25 minutes to complete this task.

*Required

Your Name: *

Please identify the defects in the given UML class diagram.

Please identify the defects in the invariants of the given UML+OCL model.

Go through the OCL constraints for the invariants corresponding to each class of the UML class diagram.

Please identify the defects in the operations of the given UML+OCL model.

Go through each operation of each class in the UML class diagram.

Submit

Never submit passwords through Google Forms.

Powered by

This content is neither created nor endorsed by Google.

Report Abuse - Terms of Service - Additional Terms

E.3.2 Flight Booking

Edit this form

VCL Defect Detection Task: Flight Booking

You were given a VCL model with seeded defects and your task is to identify them. You have 25 minutes to complete this task.

*Required

Your Name: *

Please identify the defects in the given VCL structural diagram.

Please identify the defects in the invariants of the given VCL model.

Go through the AD corresponding to each assertion that is included in the VCL strcutural diagram.

Please identify the defects in the VCL behaviour diagram

Please identify the defects in the operations of the given VCL model.

Go through each contrat and assertion diagram that is included in the behaviour diagram. Don't forget to check all included assertions and contracts in the contract and assertion diagrams as

well.

Submit

Never submit passwords through Google Forms.

Edit this form

UML+OCL. Flight Booking. Defect Detection Task

You were given a UML+OCL mode of the 'flight booking' system with seeded defects and your task is to identify them. You have 25 minutes to complete this task.

*Required

Your Name: *

Please identify the defects in the given UML class diagram.

Please identify the defects in the invariants of the given UML+OCL model.

Go through the OCL constraints for the invariants corresponding to each class of the UML class diagram.

Please identify the defects in the operations of the given UML+OCL model.

Go through each operation of each class in the UML class diagram.

Submit

Never submit passwords through Google Forms.

E.4 Model Comprehension

E.4.1 University Library

Edit this form

VCL Model Comprehension Task: University Library

You were given a correct VCL model of the University Library case study. Please answer the following questions, which try to assess how well you comprehend the given VCL model. You have 15 minutes to complete this task.

*Required

Your Name *

Please state whether the following descriptions of states of the University Library system are possible or impossible.

	Possible	Impossible
A 'one-week' 'Copy' whose status is 'toCollect' related through relation-edge 'Borrows' to a 'taughtStd' 'Member' borrowing 7 other copies	○	○
A one-month 'Copy' whose status is 'recalled' being borrowed by a 'staff' 'Member' with 25 borrowings.	○	○
A 'restricted' 'Copy' whose status is 'recalled' being borrowed by a 'researchStudent' 'Member' with 15 borrowings.	○	○
A 'Copy' with status 'shelved' and the renewals property with value '2'	○	○

Consider the global operation 'Return' and its component operations. Please state whether the following statements are true or false.

	True	False
The global 'Return' operation obtains instance of 'Book' associated with the copy number (cNo?) given as input	○	○
The operation removes the recall (association between 'Member' and 'Copy') from the	○	○

	True	False
'HasRecalled' relation		
The operation removes the borrowing (association between relevant 'Member' and 'Copy' instances) from the 'Borrows' relation given the appropriate 'Member' instance	○	○
The due date of the 'Copy' instance corresponding to the returned copy is set to the empty set	○	○
The status of the returned 'Copy' is set to 'toCollect' if the current status is 'recalled'	○	○
The status of the returned 'Copy' is set to 'shelved' if the current status is 'toCollect'	○	○

Consider a book 'Copy' that is sucessfully recalled. What are the 'before' and 'after' values of the Copy's 'status' property in the context of the 'Recall' operation?

- ○ before: 'toCollect'; after: 'recalled'
- ○ before: 'recalled; after: 'recalled'
- ○ before: 'recalled'; after: 'onLoan'
- ○ before: 'onLoan'; after: 'recalled'

What happens when local operation 'Recall' of class set 'Copy' is called on a 'one-month' copy whose status is 'toCollect'?

- ○ The operation does not run because the precondition is not satisfied as the copy's status is not 'shelved'
- ○ The copy's 'dueDate' is updated if the due date is before one week from the current time and the 'status' is set to 'recalled'
- ○ The copy's 'dueDate' is updated if the due date is before one-month from the current time and the 'status' is set to 'shelved'
- ○ None of the above

Consider a previously recalled 'Copy' that is sucessfully collected from the library. What are the 'before' and 'after' values of the Copy's 'status' property in the context of the 'collect' operation?

- ○ before: 'shelved'; after: 'onLoan'
- ○ before: 'recalled'; after: 'toCollect'
- ○ before: 'onLoan'; after: 'onLoan'
- ○ before: 'toCollect'; after: 'onLoan'

What happens when the global operation 'Renew' is called on a borrowed copy with status 'recalled' that has not been renewed before by the current taught student borrower who borrows 9 other copies?

- The operation does not run because the borrower reached the limit of borrowings
- The property 'renewals' of 'Copy' is incremented
- The operation does not run because copies with status 'recalled' cannot be renewed
- None of the above

Submit

Never submit passwords through Google Forms.

Edit this form

UML+OCL Model Comprehension Task: University Library

You were given a correct UML+OCL model of the University Library case study. Please answer the following questions, which try to assess how well you comprehend the given UML+OCL model. You have 15 minutes to complete this task.

*Required

Your Name: *

Please state whether the following descriptions of states of the University Library system are possible or impossible.

	Possible	Impossible
A 'Member' object related twice with the same 'Copy' object through associations 'Borrows' and 'HasRecalls'	○	○
A 'Member' object with category 'taughtStd' related with 10 'Copy' objects with status 'onLoan' through association 'Borrows' and 10 other 'Copy' objects with status 'recalled' through association 'HasRecalls'	○	○
A 'oneMonth' 'Copy' object with status 'shelved' related to a 'staff' 'Member' object with 15 borrowings through association 'Borrows'	○	○
A 'Copy' with 'status' 'toCollect' and the 'renewals' property with value '2'	○	○

Consider the global operation 'Recall' and its component operations. Please state whether the following statements are true or false.

	True	False
The operation obtains instance of 'Book' associated with the instance of 'Copy', corresponding to the copy	○	○

	True	False
number given as input, to check that there no copies of the book available for borrowing		
The operation obtains the instance of 'Member' from the recalled 'Copy'	○	○
The operation checks that the book associated with the recalled 'Copy' exists	○	○
The operation adds a new recall (association between 'Member' and 'Copy' instances) to the 'HasRecalled' association	○	○
The 'status' of the recalled 'Copy' is set to 'recalled' if the current status is 'toCollect'	○	○
The 'dueDate' of the recalled 'Copy' is set to a new date if the current 'dueDate' is after the re-calculated due date	○	○

What happens when operation 'Copy.return()' is called on a 'one-week' copy whose status is 'onLoan' and the 'dueDate' is '06/06/2013' for a copy that is returned on the '07/06/2013'?

- ○ The operation does not run because the precondition is not satisfied as the copy is overdue
- ○ The copy's 'dueDate' is set to the 'empty set', the 'borrower' is set to the empty set and the 'status' is set to 'shelved' and 'renewals' is set to '0'
- ○ The copy's 'dueDate' is increased by one week, the 'status' is set to 'shelved', the borrower is updated and 'renewals' is set to '0'
- ○ The copy's 'dueDate' is increased by one-week, the 'status' is set to 'onLoan', the borrower is updated and 'renewals' is set to '0'

Consider a book 'Copy' that is sucessfully collected. What are the 'before' and 'after' values of the Copy's 'status' property in the context of the collect operation?

- ○ before: 'recalled'; after: 'recalled'
- ○ before: 'toCollect'; after: 'onLoan'
- ○ before: 'toCollect; after: 'recalled'
- ○ before: 'recalled'; after: 'toCollect'

Consider a recalled book 'Copy' that is returned to the library. What are the 'before' and 'after' values of the Copy's 'status' property in the context of the return operation?

- ○ before: 'recalled'; after: 'toCollect'
- ○ before: 'recalled'; after: 'shelved'
- ○ before: 'toCollect; after: 'onLoan'
- ○ before: 'onLoan'; after: 'recalled'

Please select the best description of what is specified in the global operation 'Library.collect()'?

- The 'Member' and 'Copy' objects are retrieved from the given library and copy numbers, respectively. The operations 'Copy.recall()' and 'Member.return()' are then called on the retrieved 'Copy' and 'Member' objects
- The 'Member' and 'Copy' objects are retrieved from the given library and copy numbers, respectively. The operations 'Copy.recall()' and 'Member.renew()' are then called on the retrieved 'Copy' and 'Member' objects
- The 'Member' and 'Copy' objects are retrieved from the given library and copy numbers, respectively. The operations 'Copy.recall()' and 'Member.recall()' are then called on the retrieved 'Copy' and 'Member' objects
- None of the above

Submit

Never submit passwords through Google Forms.

E.4.2 Flight Booking

Edit this form

VCL Model Comprehension Task: Flight Booking

You were given a correct VCL model of the Flight Booking case study. Please answer the following questions, which try to assess how well you comprehend the given VCL model. You have 15 minutes to complete this task.

*Required

Your Name *

Please state whether the following descriptions of states of the Flight Booking system are possible or impossible.

	Possible	Impossible
A reservation with the fare given in currency 'USD' and the penalty given in currency 'EUR'	○	○
A 'Reservation' object that is connected, via a 'Seat', to a 'Flight', that has already made its journey.	○	○
A 'Reservation' object with status 'void' that is connected to a 'Seat' through association 'HasReservation'	○	○

Consider the global operation 'Reserve' and its component operations. Please state whether the following statements are true or false.

	True	False
The operation obtains the instance of 'Flight' that is associated with the flight number given as input using the operation 'GetFlightFrNoAndDate'	○	○
The operation checks that the flight's destination is consistent with the destination selected by the passenger using operation 'IsConsistentDestination'	○	○

	True	False
The operation checks that the current date is before the date of the intended flight to reserve	○	○
The operation obtains a free seat for the selected flight given the seat class and class of fare selected by the passenger	○	○
The operation creates a reservation and associates, through association 'HasReservation', the newly created 'Reservation' instance with the 'Flight' instance corresponding to the selected flight using operation 'AddReservationForSeat'	○	○

What happens when the global operation 'CancelReservation' is called to cancel a 'flex' reservation whose status is 'confirmed'?

○ The operation does not run because the precondition is not satisfied as the status of the 'Reservation' is 'confirmed'

○ The 'status' of the 'Reservation' instance is set to 'void' and the link between the 'Seat' and the 'Reservation' is removed

○ The 'status' of the 'Reservation' instance is set to 'pending' and the link between the 'Seat' and the 'Reservation' is removed

○ The 'status' of the 'Reservation' instance is set to 'void', the link between the 'Seat' and the 'Reservation' is removed and the reserved 'Seat' is set to status 'free'

What exactly happens when the operation 'changeReservationFlightAndDate' is called on the day before the reserved flight's departure date on a 'RelaxPlus' 'Reservation' to change to an 'economy' 'Seat' of a flight provided by an airline that is different from the airline of the original reservation?

○ The operation does not run because the pre-condition is not satisfied as there is not enough difference between the date of changing the reservation and the flight's departure date

○ The operation does not run because the pre-condition is not satisfied as the airline is different.

○ The reservation is made to another free seat of the flight corresponding to the given flight number. The 'status' of the newly reserved 'Seat' instance becomes 'booked' and the previous one becomes 'free'. The 'Reservation' instance, which was linked to the previous 'Seat' instance, becomes linked to the new 'Seat' instance.

○ The reservation is made to another free seat of the flight corresponding to the given flight number. The 'status of the newly reserved 'Seat' instance becomes 'booked' and the previous one becomes 'free'. The fare associated with the 'Reservation' is updated to reflect the reservation's penalty.

Consider instances of 'Reservation' and 'Seat' that are affected by the operation of successfuly cancelling a reservation. What are the 'before' and 'after' values of the status'

properties of the involved instances of 'Reservation' and 'Seat' in the context of this operation?

- Reservation': before anything other than 'void', after 'void'. 'Seat': before 'booked', after 'free'.
- Reservation': before 'pending', after 'void'. 'Seat': before 'free', after 'booked'.
- Reservation': before 'confirmed', after 'void'. 'Seat': before 'booked', after 'free'.
- Reservation': before 'hold', after 'void'. 'Seat': before 'free', after 'booked'.

What happens when the global operation 'ChangeReservationDate' is called in 'Reference' to a 'Reservation' with status 'confirmed' and fare class 'saver' one day before the flight's departure date?

- The operation does not run because the pre-condition is not satisfied as there is not enough difference between the date of changing the reservation and the flight's departure date
- The reservation changes the 'status' to 'pending', the link with the previously reserved 'Seat' is removed, and there is a new link to a free seat of the reserved flight on the selected date.
- The reservation remains with 'status' confirmed, the link with the previously reserved 'Seat' is removed, and there is a new link to a free seat of the reserved flight on the selected date.
- None of the above

Submit

Never submit passwords through Google Forms.

Edit this form

UML+OCL Model Comprehension Task: Flight Booking

You were given a correct UML+OCL model of the University Library case study. Please answer the following questions, which try to assess how well you comprehend the given UML+OCL model. You have 15 minutes to complete this task.

*Required

Your Name: *

Please state whether the following descriptions of states of the Flight Booking system are possible or impossible.

	Possible	Impossible
A 'Reservation' instance with status 'pending' related, bidirectionally through association 'HasReservation, with a 'Seat' instance with status 'free'	○	○
A 'Seat' object with class 'EconomyPlus' and with the 'Flex' class fare as one of its available class fares (property 'Seat.fares')	○	○
Two distinct instances of class 'Airline' that operate flights from the same instance of class 'Airport' and that have, therefore, most flights originating from this airport.	○	○
A 'Reservation' with 'reservationDate' that is the same as the date of the reserved 'Flight'	○	○

Please consider the operation 'FlightBoooking.changeReservationDate()' and its component operations. Please state whether the following statements concerning this operation are true or false.

	True	False
This operation tries to find the 'Reservation' object that is	○	○

	True	False
associated with parameter 'rNo' through operation 'getReservationFromNo'		
This operation checks that there is a reservation for the given reservation number ('rNo') using function 'FlightBooking.hasReservation()'	○	○
The operation checks that the new flight date is before the date when the original reservation was made using function 'Date.IsBefore()'	○	○
This operation unbooks the 'Seat' object that is currently associated with the 'Reservation', books another 'Seat' object with status 'free' that is to be associated with the reservation and updates the reservation's fare	○	○
The operation tries to find a 'Seat' instance with status 'free' for the same flight number and class of the original reservation given some date using the function 'FlightBooking.getFreeSeatForFlight()'	○	○
The reservation's previous seat is unbooked using operation 'Seat.unBook()', which is called from 'FlightBooking.changeReservationDate()'	○	○

What happens when the local operation 'changeReservationFlightAndDate' of class 'Reservation' is called on a 'relax' 'Reservation' instance with status 'confirmed'?

○ The operation does not run because the precondition is not satisfied as the status of the 'Reservation' is 'confirmed'

○ The 'fare' property of the 'Reservation' instance is updated.

○ The operation does not run because the precondition is not satisfied as the status of the 'Reservation' is 'relax'

○ The class of the seat is downgraded to 'economy'

What happens when the operation 'changeReservationDate' is called on the same day of the reserved flight's departure date, but 20 hours before the flight's departure, to change a class 'economy' seat reserved with a 'flex' fare?

○ The operation does not run because the pre-condition is not satisfied as there is no difference between reservation changing and flight dates

○ The reservation is made to another seat for the selected flight date and for a flight with the same flight number as the original reservation

- The reservation is made to another seat for the selected flight date and selected flight
- The operation does not run because the pre-condition is not satisfied as the class fare is 'flex'

Consider instances of 'Reservation' and 'Seat' that are affected by the operation of successfully reserving a seat. What are the 'before' and 'after' values of the 'status' properties in the involved instances of 'Reservation' and 'Seat'?

- 'Reservation': before 'pending', after 'hold'. 'Seat': before 'free', after 'booked'.
- Reservation': before 'pending', after 'hold'. 'Seat': before 'booked', after 'free'.
- 'Reservation': before 'hold', after 'confirmed'. 'Seat': before 'free', after 'booked'.
- 'Reservation': no before, after 'pending'. 'Seat': before 'free', after 'booked'.

Consider instances of 'Reservation' and 'Seat' that are affected by the operation of successfully changing the date of a reservation. What are the 'before' and 'after' values of the 'status' properties in the involved instances of 'Reservation' and 'Seat'?

- 'Reservation': before 'confirmed', after 'confirmed'. 'Seat': before 'free', after 'booked'.
- 'Reservation': before 'hold', after 'hold'. 'Seat': before 'free', after 'booked'.
- 'Reservation': before 'confirmed', after 'confirmed'. Previous 'Seat': before 'booked', after 'free'. New 'Seat': before 'free', after 'booked'.
- 'Reservation': before 'hold', after 'hold'. Previous 'Seat': before 'booked', after 'free'. New 'Seat': before 'free', after 'booked'.

Submit

Never submit passwords through Google Forms.

E.5 Debriefing

Edit this form

VCL vs UML+OCL. Debriefing Survey

This is the debriefing survey form of the VCL vs UML+OCL experiment. It is designed to collect information on the personal experiences, impressions and outcomes from the experiment's participants.

Please respond to each question as thoughtfully and honestly as you can. Anything you write down will be treated confidentially.

Thank you.

*Required

Your Name: *

Experiment's venue *

The Experiment

The following questions are about your overall impression of the experiment in general.

Please indicate the extent to which you agree or disagree with the following statements, concerning your general impressions, feelings and personal outcomes coming from your participation in the experiment. *

Levels of agreement: 1: Strongly agree; 2: Agree; 3: Uncertain; 4: Disagree; 5: Strongly disagree

	1	2	3	4	5
I liked the experiment					
The experiment was interesting					
During the experiment, I was motivated, committed and I felt challenged					
I felt it was a god learning experience					

Plese state the reasons for your lack of motivation concerning the experiment, if you felt somehow unmotivated

Tick all that apply

- [ ] I do not really see the need for software design modelling in software engineering
- [ ] I do not know enough about software design modelling
- [ ] I could not master the modelling languages under study
- [ ] The experiment's tasks were not very interesting
- [ ] I did not fully understand the experiment's tasks
- [ ] I did not feel comfortable with the modelling tools
- [ ] Other:

Please use your own words to comment on your general experiment participation experience?

Training

The following questions are about the training that was given to you as part of the experiment.

Please say the extent to which you agree or disagree with the following statements concerning the training that was given to you. *

Levels of agreement: 1: Strongly agree; 2: Agree; 3: Uncertain; 4: Disagree; 5: Strongly disagree

	1	2	3	4	5
The training allowed me to carry out the tasks of the experiment competently	○	○	○	○	○
I had enough training in UML	○	○	○	○	○
I had enough training in OCL	○	○	○	○	○
I had enough training in VCL	○	○	○	○	○
I would have performed better if I were given more hours of training	○	○	○	○	○

Please use your own words to comment on the training that was given to you, if you feel necessary?

Experiment's Tasks

The following are general questions about the tasks of the experiment.

With respect to your general experience of carrying out the experiment's tasks, please say the extent to which you agree or disagree with the following statements. *

Levels of agreement: 1: Strongly agree; 2: Agree; 3: Uncertain; 4: Disagree; 5: Strongly disagree

	1	2	3	4	5
I had enough time to read through the requirements descriptions of the given case studies	○	○	○	○	○
I fully understood the systems described in the requirements documents	○	○	○	○	○
In general, I had no problems carrying out the experiment's tasks	○	○	○	○	○
The instructions were clear and easy to follow	○	○	○	○	○
The university library system was fairly easy to understand	○	○	○	○	○
The flight booking system was fairly easy to understand	○	○	○	○	○
The university library case study was realistic; a good example of a real-world case study	○	○	○	○	○

	1	2	3	4	5
The flight booking case study was realistic; a good example of a real-world case study	○	○	○	○	○
The modelling tools were fairly stable	○	○	○	○	○

Based on your experience, which of the two case studies used in the experiment is more complex?

- ○ I felt that the University Library case study is more complex than the Flight Booking case study
- ○ I felt that the Flight Booking case study is more complex than the University Library case study
- ○ I think that both case studies have a similar complexity

Please indicate the factors that hindered your performance in the experiment.

Please tick all that apply

- ☐ I often ran out of time
- ☐ I did not fully understand the tasks
- ☐ I did not fully understand the case studies
- ☐ I felt I did not have enough training
- ☐ I felt I lacked knowledge and experience in software design modelling
- ☐ I felt tired
- ☐ My mind was somewhere else
- ☐ Other:

The modelling tasks

Each working session started with two modelling tasks for a given case study. The first involved defining the state space and associated invariants The second involved defining part of the behaviour. The following enquires about your personal modelling experience.

With respect to the tasks that involved the construction of design models with the languages under study, please indicate the extent to which you agree or disagree with the following statements. *

Levels of agreement: 1: Strongly agree; 2: Agree; 3: Uncertain; 4: Disagree; 5: Strongly disagree

	1	2	3	4	5
I had no problems defining the state structures that make the system's state	○	○	○	○	○

	1	2	3	4	5
space					
I had no problems modelling the invariants	○	○	○	○	○
I had no problems modelling the behavior of the given case studies	○	○	○	○	○
I was not sure about what I was supposed to model	○	○	○	○	○
The case studies were too complex	○	○	○	○	○
The instructions were very clear to me	○	○	○	○	○
The time given was enough for me to execute the modelling tasks comfortably	○	○	○	○	○

Please indicate the factors that hindered your performance in the modelling tasks.

Tick all that apply

- ☐ Lack of time
- ☐ Lack of knowledge and experience in software design modelling
- ☐ The case studies were complex
- ☐ Lack of knowledge and experience with the modelling languages
- ☐ Lack of experience with the modelling tools
- ☐ Not enough training
- ☐ The tasks were not very clear
- ☐ Other:

Please indicate the modelling language in which you felt you performed better, VCL or UML+OCL, for the following modelling tasks. *

	VCL	UML+OCL	No Preference
Modelling the state space structures (sets, classes, datatypes, etc in UML class diagrams or VCL structural diagrams)	○	○	○

	VCL	UML+OCL	No Preference
Modelling invariants	○	○	○
Defining operation signatures (in UML class diagrams or VCL behaviour diagrams)	○	○	○
Defining actual behaviour of operations using pre- and post-conditions	○	○	○

Please indicate the language in which you performed better in the modelling tasks, if you feel you have performed better using one language than the other. *

- ○ VCL
- ○ UML+OCL
- ○ No difference

Please use your own words to justify your answers to the last three questions. *

Please provide further comments concerning your experience of modelling the given case studies using the languages under study, if you feel necessary.

The Problem Comprehension Task

After the modelling tasks, you were asked to complete a questionnaire that tried to evaluate your comprehension of the given problem. The following enquires about your experience in undertaking this task.

Please indicate the extent to which you agree or disagree with the following statements. *

Levels of agreement: 1: Strongly agree; 2: Agree; 3: Uncertain; 4: Disagree; 5: Strongly disagree

	1	2	3	4	5
I felt I performed well in the problem comprehension questionnaires	○	○	○	○	○
I could understand the functionalities of the systems from the requirements descriptions	○	○	○	○	○
The modelling task helped me in comprehending the problem	○	○	○	○	○
Some questions were either unclear or ambiguous	○	○	○	○	○
Some questions were difficult	○	○	○	○	○
The time given was enough for me to execute this task comfortably	○	○	○	○	○

Please indicate the main factor that contributed towards your comprehension of the problem. *

- ○ The modelling activity
- ○ The requirements text
- ○ After having modelled and worked on the problem so many times using boh VCL and UML+OCL
- ○ Other: _____

Please indicate the factors that hindered your performance in the problem comprehension task.

Tick all that apply

- ☐ Lack of time
- ☐ Lack of knowledge and experience in software design modelling
- ☐ The case studies were complex
- ☐ Underperformance in the modelling task
- ☐ I did not fully understand the task
- ☐ The instructions were not very clear
- ☐ Unclear or ambiguous questions
- ☐ Difficult questions

☐ Not enough training

☐ Other:

Did you feel that any of the languages under study (VCL or UML+OCL) posed an advantage over the other in terms of problem comprehension? *

○ I felt that modelling in UML+OCL was a better aid to problem comprehension than modelling in VCL

○ I felt that modelling in VCL was a better aid to problem comprehension than modelling in UML+OCL

○ I didn't feel an advantage of either language in terms of aid to problem comprehension

Please use your own words to justify you answer to the previous question. *

Please comment on your experience of answering problem comprehension questionnaires, if you feel necessary.

The Defect Detection Task

After the problem comprehension task, you were asked to detect errors in a given model. The following enquires about your personal experience of this task.

With respect to your experience in the defect detection tasks, please indicate the extent to which you agree or disagree with the following statements. *

Levels of agreement: 1: Strongly agree; 2: Agree; 3: Uncertain; 4: Disagree; 5: Strongly disagree

	1	2	3	4	5
I felt I performed well in the defect detection tasks	○	○	○	○	○
The instructions were clear	○	○	○	○	○
I fully understood the task	○	○	○	○	○

	1	2	3	4	5
I was not sure what type of errors I was looking for	○	○	○	○	○
The defect detection task helped me in comprehending what the systems were trying to do	○	○	○	○	○
The time given was enough for me to execute this task comfortably	○	○	○	○	○

Please indicate the factors that hindered your performance in the defect detection task.

Tick all that apply

- ☐ Lack of time
- ☐ Lack of knowledge and experience with the modelling languages under study
- ☐ The case studies were complex
- ☐ I did not fully understand the task
- ☐ Poor performance in the modelling task
- ☐ Difficulty in understanding OCL Expressions
- ☐ Difficulty in understanding VCL Expressions
- ☐ Uneasiness with the modelling tools
- ☐ Not enough training
- ☐ Other:

Did you feel that any of the languages under study (VCL or UML+OCL) posed an advantage over the other in terms of the defect detection task? *

- ○ I felt that it was easier to find defects in UML+OCL models than VCL ones
- ○ I felt that it was easier to find defects in VCL models than UML+OCL ones
- ○ I didn't feel that either VCL or UML+OCL posed an advantage in terms of defect detection

Did you feel that any of the tools supporting the languages under study (VCB or Papyrus UML) provided an advantage over the other for the defect detection tasks? *

- ○ I felt that the defect detection task was easier with Papyrus UML
- ○ I felt that the defect detection task was easier with VCB
- ○ I didn't feel that either of the tools posed an advantage in defect detection

Please use your own words to justify you answers to the two previous questions. *

Please provide further comments concerning your experience in the defect detection tasks with the languages under study, if you feel necessary.

The Model Comprehension Task

After the defect detection task, you were given a complete model of one of the case studies and you were asked to complete a questionnaire concerning your comprehension of the given model. This was a task to evaluate your interpretation of a given model.

The following enquires about this task.

Please indicate the extent to which you agree or disagree with the following statements, concerning your experience in the model comprehension task. *

Levels of agreement: 1: Strongly agree; 2: Agree; 3: Uncertain; 4: Disagree; 5: Strongly disagree

	1	2	3	4	5
I felt I performed well					
The task was very clear to me					
The instructions were clear					
The task helped me in comprehending the functionality of the modelled system and its model					
Some questions were either unclear or ambiguous					
Some questions were difficult					

	1	2	3	4	5
The time given was enough for me to execute this task comfortably	○	○	○	○	○

Please indicate the factors that hindered your performance in the model comprehension task. *

Tick all that apply

- ☐ Lack of time
- ☐ Lack of knowledge and experience with the modelling languages under study
- ☐ The case studies were complex
- ☐ The instructions were not clear
- ☐ I did not fully understand the task
- ☐ Poor performance in modelling and defect detection
- ☐ Unclear or ambiguous questions
- ☐ Difficult questions
- ☐ Not enough training
- ☐ Other:

Did you feel that any of the languages under study (VCL or UML+OCL) posed an advantage over the other in terms of model comprehension? *

- ○ I felt that the model comprehension task was easier with UML+OCL
- ○ I felt that the model comprehension task was easier with VCL
- ○ I didn't feel that either VCL or UML+OCL posed an advantage

Did you feel that any of the tools supprting the languages under study (VCB or Papyrus UML) provided an advantage over the other for the model comprehension tasks? *

- ○ I felt that the model comprehension task was easier with Papyrus UML
- ○ I felt that the model comprehension task was easier with VCB
- ○ I didn't feel that either of the tools posed an advantage in the model comprehension tasks

Please use your own words to justify you answers to the previous two questions. *

Please provide any further comments concerning your experience in the model comprehension tasks with the languages under study, if you feel necessary.

Usefulness

The questions that follow try to assess your perceived usefulness of the languages and tools used in the experiment.

Perceived usefulness can be classified as the degree to which a person believes that using a particular system would enhance his or her performance in some task.

Based on your experience of using UML+OCL in the experiment, please indicate the extent to which you agree or disagree with the following statements concerning usefulness *

Levels of agreement: 1: Strongly agree; 2: Agree; 3: Uncertain; 4: Disagree; 5: Strongly disagree

	1	2	3	4	5
UML+OCL helps me being more effective in my modelling tasks	○	○	○	○	○
UML+OCL helps me being more productive in my modelling tasks	○	○	○	○	○
UML+OCL enhances my performance in modelling related activities	○	○	○	○	○
UML+OCL is not useful	○	○	○	○	○
UML+OCL does not meet my needs	○	○	○	○	○
UML+OCL does everything I would expect it to do	○	○	○	○	○
UML+OCL does not help me to have a better control over software modelling and the process of software	○	○	○	○	○

	1	2	3	4	5
development in general					
I am not satisfied with UML+OCL	○	○	○	○	○
I would not recommend UML+OCL to other people	○	○	○	○	○
If well used, UML+OCL can improve the quality of software development	○	○	○	○	○
If well used, UML+OCL can improve productivity in software development	○	○	○	○	○

Based on your experience of using VCL in the experiment, please indicate the extent to which you agree or disagree with the following statements concerning usefulness *

Levels of agreement: 1: Strongly agree; 2: Agree; 3: Uncertain; 4: Disagree; 5: Strongly disagree

	1	2	3	4	5
VCL helps me being more effective in my modelling tasks	○	○	○	○	○
VCL helps me being more productive in my modelling tasks	○	○	○	○	○
VCL enhances my performance in modelling related activities	○	○	○	○	○
VCL is not useful	○	○	○	○	○
VCL does not meet my needs	○	○	○	○	○
VCL does everything I would expect it to do	○	○	○	○	○
VCL does not help me to have a better control over software modelling and the process of	○	○	○	○	○

	1	2	3	4	5
software development in general					
I am not satisfied with VCL	○	○	○	○	○
I would not recommend VCL to other people	○	○	○	○	○
If well used, VCL can improve the quality of software development	○	○	○	○	○
If well used, VCL can improve the productivity in software development	○	○	○	○	○

Based on your experience in the experiment, please indicate the language that you feel is more useful for the following modelling-related activities

	VCL	UML+OCL	No difference
Modelling the state space	○	○	○
Modelling the state space constraints using invariants	○	○	○
Modelling operations based on design-by contract	○	○	○
Activities that involve using design models, such as defect detection	○	○	○

Which of the languages under study, VCL or UML+OCL, do you perceive as more useful? *

- ○ I think UML+OCL is more useful
- ○ I think VCL is more useful
- ○ I think both languages are equally useful

Please justify your answers to the previous two questions *

Ease of use

The questions of this section try to assess your perceived ease of use of the languages used in the experiment.

Perceived ease of use can be classified as the degree to which a person believes that using a particular system would be free of effort.

Based on your experience of using UML+OCL in the experiment, please indicate the extent to which you agree or disagree with the following statements concerning ease of use *

Levels of agreement: 1: Strongly agree; 2: Agree; 3: Uncertain; 4: Disagree; 5: Strongly disagree

	1	2	3	4	5
UML+OCL is not user friendly	○	○	○	○	○
UML+OCL is frustrating	○	○	○	○	○
UML+OCL is not flexible	○	○	○	○	○
UML+OCL is cumbersome	○	○	○	○	○
UML+OCL is difficult to use	○	○	○	○	○
UML+OCL cannot be used without training or written instructions	○	○	○	○	○
UML+OCL is not easy to learn	○	○	○	○	○
UML+OCL is clear and understandable	○	○	○	○	○
UML+OCL is not intuitive	○	○	○	○	○
UML+OCL models are confusing	○	○	○	○	○
UML +OCL models are difficult to understand and interpret	○	○	○	○	○
Navigation in UML+OCL is consistent and	○	○	○	○	○

	1	2	3	4	5
intuitive					
Error messages are easy to understand and respond to	○	○	○	○	○
Repetitive actions and frequent activities can be made easier	○	○	○	○	○
UML+OCL's hidden dependencies complicate model interpretation	○	○	○	○	○
UML+OCL does not make software modelling more pleasant	○	○	○	○	○

Based on your experience of using VCL in the experiment, please indicate the extent to which you agree or disagree with the following statements concerning ease of use *

Levels of agreement: 1: Strongly agree; 2: Agree; 3: Uncertain; 4: Disagree; 5: Strongly disagree

	1	2	3	4	5
VCL is not user friendly	○	○	○	○	○
VCL is frustrating	○	○	○	○	○
VCL is not flexible	○	○	○	○	○
VCL is cumbersome	○	○	○	○	○
VCL is difficult to use	○	○	○	○	○
VCL cannot be used without training or written instructions	○	○	○	○	○
VCL is not easy to learn	○	○	○	○	○
VCL is clear and understandable	○	○	○	○	○
VCL is not intuitive	○	○	○	○	○
VCL models are confusing	○	○	○	○	○
Navigation in VCL is consistent and intuitive	○	○	○	○	○

	1	2	3	4	5
VCL models are difficult to understand and interpret	○	○	○	○	○
Error messages are easy to understand and respond to	○	○	○	○	○
Repetitive actions and frequent activities can be made easier	○	○	○	○	○
VCL's hidden dependencies complicate model interpretation	○	○	○	○	○
VCL does not make software modelling more pleasant	○	○	○	○	○

Based on your experience in the experiment, please indicate the language that you feel is more easy to use for the following modelling-related activities

	VCL	UML+OCL	No difference
Modelling the state space	○	○	○
Modelling the state space constraints using invariants	○	○	○
Modelling operations based on design-by contract	○	○	○
Activities that involve using design models, such as defect detection	○	○	○

Which of the languages under study, VCL or UML+OCL, do you perceive as more easy to use? *

- ○ I think UML+OCL is more easy to use
- ○ I think VCL is more easy to use
- ○ I think both languages are equally easy to use

Please justify your answer to the previous question *

Final Comparisons

The questions that follow make some direct comparisons between the modelling languages under study.

Based on your experiment experience, please indicate the language and tool that in your opinion perform better at the following features. *

	VCL	UML+OCL	No preference
Readability	○	○	○
Navigation through model artifacts	○	○	○
Maps and overviews	○	○	○
Live error checking	○	○	○
Look and feel	○	○	○
Cohesion and integration between the different parts	○	○	○
Learnability	○	○	○
Comfort/Satisfaction	○	○	○
Complexity Management (representation of information without overloading human mind)	○	○	○
Perceptual discriminability/Visibility (how elements are distinguished from each other)	○	○	○
Engagement	○	○	○

Please justify your answers to the two previous questions. *

Please indicate your prefered notation from either the UML+OCL or VCL suite. *

	UML+OCL	VCL	No preference
UML class diagrams vs VCL Structural diagrams	○	○	○
OCL constraints vs VCL assertion diagrams	○	○	○
OCL constraints for pre- and post-conditions vs VCL Contract Diagrams	○	○	○
UML class diagrams for operations vs VCL behaviour diagrams	○	○	○

Please justify your answer to the previous question. *

Thinking about your experience in the experiment and the modelling languages that you've used. Overall, what was your preferred modelling language? *

- ○ UML+OCL
- ○ VCL
- ○ No preference

Thinking about your experience in the experiment and the modelling tools that you've used. Overall, what was your preferred modelling tool? *

- ○ Papyrus UML
- ○ VCB
- ○ No preference

If you had to choose one of the languages under study for future developments of software design models, which would be the language of your choice? *

- ○ UML+OCL
- ○ VCL
- ○ No preference

Please justify your answer to the three previous questions. *

Please provide any further comments concerning your participation in the experiment, if you feel necessary.

Submit

Never submit passwords through Google Forms.