Empirical Software Engineering manuscript No.
(will be inserted by the editor)

An Experimental Scrutiny of Visual Design Modelling: VCL up against UML+OCL

Nuno Amálio · Lionel Briand · Pierre Kelsen

the date of receipt and acceptance should be inserted later

Abstract The graphical nature of prominent modelling notations, such as the standards UML and SysML, enables them to tap into the cognitive benefits of diagrams. However, these notations hardly exploit the cognitive potential of diagrams and are only partially graphical with invariants and operations being expressed textually. The Visual Contract Language (VCL) aims at improving visual modelling; it tries to (a) maximise diagrammatic cognitive effectiveness, (b) increase visual expressivity, and (c) level of rigour and formality. It is an alternative to UML that does largely pictorially what is traditionally done textually. The paper presents the results of a controlled experiment carried out four times in different academic settings and involving 43 participants, which compares VCL against UML and OCL and whose goal is to provide insight on benefits and limitations of visual modelling. The paper's hypotheses are evaluated using a crossover design with the following tasks: (i) modelling of state space, invariants and operations, (ii) comprehension of modelled problem, (iii) detection of model defects and (iv) comprehension of a given model. Although visual approaches have been used and advocated for decades, this is the first empirical investigation looking into the effects of graphical expression of invariants and operations on modelling and model usage tasks. Results suggest VCL benefits in defect detection, model comprehension, and modelling of operations, providing some empirical evidence on the benefits of graphical software design.

Keywords Design · modelling · UML · OCL · VCL · Diagrams · Experiment

Nuno Amálio
Faculty of Computing, Engineering and the Built Environment
Birmingham City University, Millennium Point, Curzon Street
Birmingham, B4 7XG, UK, E-mail: nuno.amalio@bcu.ac.uk

Lionel Briand
EECS department, University of Ottawa, Canada
and SnT Center, University of Luxembourg, 29, Avenue J. F. Kennedy
L-1855 Luxembourg, E-mail: lbriand@uottawa.ca, lionel.briand@uni.lu

Pierre Kelsen
Faculty of Science, Technology and Communication, University of Luxembourg
Maison du Nombre, 6, Avenue de la Fonte, L-4364 Esch-sur-Alzette, Luxembourg
E-mail: pierre.kelsen@uni.lu

1 Introduction

There is a crying need for modelling in software and systems engineering. Modelling no longer needs to be advocated as good engineering practice; instead, its necessity emerges from the practice of software and systems engineering. Modelling and design are simply the natural response to a need for abstraction and better means for tackling complexity.

Visual modelling has always been part of software engineering [31,96,97] with graphical design being advocated as a way forward for nearly three decades [58, 59]. Popular notations of software and systems engineering, such as the standards UML and SysML [84], are graphical, which enables them to tap into the cognitive benefits that diagrams are known to provide [70]; their graphical nature is seen as a major factor behind their popularity[1]. However, they fail to fully exploit the visual spectrum; they are criticised for having low cognitive effectiveness [79,77].

Mainstream visual notations are criticised for their semantics problems [110,42, 105,61,60,49,74,99,76]. Their semantics is seen as being flimsy and ill-defined, resulting in imprecision due to the many ways in which phrases of a language may be interpreted and misinterpreted. Models expressed in these notations may have inconsistencies in them [69,45], are prone to ambiguities and misunderstandings, and are often not mechanisable because they lack means of semantic inference. This, in turn, hampers exhaustive verifiability using theorem proving or model-checking as this requires a formally-defined semantics. Nevertheless, rigorous-minded research communities have learnt to overcome this issue; standards UML and SysML, seen by many as a family of modelling languages [35,32], are often seen as bloated language definitions that hinder rigorous usage. However, by defining well-founded profiles — a specialisation or a subset of the standard notation, or a family member —, it is possible to tackle the semantics problem; it is in this way that many formalisations of UML [109,106,19,17,18,4], OCL [95,94] and SysML [16] have been achieved.

The graphics of mainstream languages can be disappointing. They cannot express all the required properties diagrammatically; UML provides the textual OCL notation to express constraints of operations and invariants. UML's graphical description of behaviour has been criticised [60,49,74,40]; collaboration and sequence diagrams are but partial behavioural descriptions as they describe scenarios; state and activity diagrams provide total descriptions, but their emphasis on explicitly defined states and state transitions may result in descriptions that are cumbersome, especially with systems other than those traditionally classified as reactive.

The Visual Contract Language (VCL) [9,12,15,10] expresses software designs formally and graphically. It embodies a critique of the UML; it aims to improve the following: (a) diagrammatic expressivity by describing pictorially what is described textually in the UML realm; (b) visual effectiveness by designing VCL according to theories and guidelines for the design of visual notations; and (c) the semantics issue by designing VCL with a formal semantics. To achieve this, VCL comes with two novel visual notations: assertion and contract diagrams, key ingredients in ensuring that VCL does visually what is done textually with UML.

VCL's foundational premise is that graphical software design is a good idea. The paper investigates this premise to gauge the effectiveness of visual software

[1] The sequel uses the terms visual, pictorial, graphical and diagrammatic interchangeably.

design and VCL's novel diagram types. The empirical investigation involves a controlled experiment, carried out four times in university settings with 43 participants, which compares VCL against the UML and OCL standards (our baseline) to provide insight into benefits and limitations of visual modelling. The paper examines the following: (i) modelling of state space, invariants and operations, (ii) comprehension of modelled problem, (iii) detection of model defects and (iv) end-user comprehension of a given model, (v) usefulness, (vi) ease of use, (vii) usability and (viii) overall appraisal of examined notations. Results suggest benefits of VCL's graphical approach in defect detection and model comprehension, and more general benefits of visual design.

1.1 Contributions

The paper's contributions are as follows:

- This is the first empirical study that compares a design language that expresses predicates graphically against the UML and its OCL satellite notation.
- This is the first empirical demonstration suggesting benefits of a diagrammatic approach to the modelling of operations using design-by-contract [75].
- This is the first empirical demonstration that suggests that a diagrammatic modelling approach benefits tasks associated with the usage of models. The paper shows how VCL was significantly better when compared to UML+OCL in tasks related to end-user model comprehension and defect detection.

1.2 Outline

The remainder of this paper starts with some background (section 2) focussed on VCL and diagrammatic description. This is followed by the experiment's scope (section 3), design (section 4) and materials (section 5). The paper then talks about the pursued statistical analysis (section 6), the experiment's participants (section 7) and the results of the controlled experiment (section 8). Finally, it discusses threats to the validity of the paper's results (section 9), presents related work (section 10) and draws its conclusions (section 11).

2 Background

The sequel provides background on the cognitive benefits of diagrams and VCL, discusses the principles of graphical notations applied in VCL's design, and presents VCL's trajectory and founding ideas explaining why VCL is a suitable representative of graphical modelling.

2.1 Diagrams and their Cognitive Effectiveness

When solving problems, humans use both internal representations, stored in their brains, and external representations, recorded on paper or some other medium [70]. For cognitive scientists, the form of external representations matters to the agents'

performance in various cognitive tasks, such as problem-solving or decision-making [70,117]; form determines what information can be perceived, what cognitive processes can be activated and what can be discovered. According to Zhang [117], "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 favour pictorial representations [88,70,52,53,58,30]. Larkin and Simon's study [70] show that text and diagrams containing the same information are not necessarily equivalent with respect to the processing required to extract information, highlighting that diagrams facilitate perceptual inferences — some are called *free rides* because they are easily perceptible [101]. Subsequent studies [52, 53] have replicated the findings of [70], highlighting a picture advantage. However, the superiority of pictures should not be taken for granted [70,86].

The practical relevance of diagrams is endorsed by their ubiquity in engineering [46,47]. Visual thinking is seen as intrinsic to engineering; thinking, designing and communicating with pictures are recognised as essential engineering activities. Like in traditional engineering, diagrams became an integral part of software engineering [79]. Diagrams in all forms and shapes, from formal representations to informal and ephemeral sketches, constitute a prominent means of software engineering expression. Unlike traditional engineering, however, software diagrams are not tied to the physical shape of any designed artefact, which entailed greater freedom regarding diagrammatic shapes and forms [25].

Unsurprisingly, software engineering visual languages have been advocated for decades as means to facilitate human communication and problem solving [58]. Visual languages, such as Harel's statecharts [57], UML and SysML, are widely taught. The UML development is seen as a pivotal in software engineering, providing a common unifying language that the field had never had before. Although diagrams favour perceptual inferences, this does not entail cognitive effectiveness [70, 86] as diagrams need to be designed to exploit the benefits of visual representations [70,79]. This is where mainstream visual languages lag behind; the UML, for instance, is criticised for breaching many rules of visual language design [79,77].

2.2 A Primer on VCL

VCL follows the approach to modelling of UML and many of UML's predecessors [98,26,110] based on an object oriented (OO) style of description, which sees a software system as a collection of data with associated behaviour inter-operatable from the environment. Another VCL pillar is discrete mathematics and set theory, the foundation of languages such as Z [103,114,64] and B[1].

VCL is introduced here with an example of a secure bank whose complete VCL model is given in [5]. A banking system manages customers, accounts and transactions, and needs to be made secure. Figures 1 and 2 present a few model excerpts that illustrate VCL.

VCL organises models around inter-dependent packages using ideas from aspect orientation [15]. Figures 1 and 2 present diagrams of two different packages. Package diagrams (PDs) define packages (represented as clouds) and their dependencies to other packages. PD of Fig. 1a says that package `Bank` imports sets from `CommonTypes`. Package `CommonTypes` provides definitions common across the

model; Bank focuses on those concerns specific to banking; other packages in [5] address security concerns and the modular weaving of security and banking.

Structural diagrams (SDs), VCL's UML class diagram equivalent, express domains of discourse in state space (data or conceptual) models by representing entities of interest as sets (round contours). SD of Bank (Fig. 1b) has class sets for banking entities, namely, customers, accounts and transactions, and uses value sets from SD of CommonTypes (Fig. 1c) — e.g. CT::Name. Objects and values are depicted as rectangles. Sets for dates and times of Fig. 1c use VCL's predicate language to define the novel *derived sets*. Set Month, for instance, is defined as the natural numbers (set Nat) from 1 to 12 in a graphical depiction of a *set comprehension* — $Month = \{n : Nat \mid n \geq 1 \wedge n \leq 12\}$. The arrows emanating from Nat (*predicate edges*) refer to the source and are combined through conjunction.

State spaces are subject to constraints (or *invariants*) that must be respected in all states of the system. Unlike UML, VCL identifies invariants in the data model — a SD defines a state space made up of state structures and their constraints — as *assertions* (named elongated hexagons) defined in assertion diagrams (ADs). In Fig. 1b, assertions HasCurrentBefSavings, SavingsArePositive and CorporateHaveNoSavings, defined in ADs of Figs. 1d, 1e and 1f, embody relevant invariants: customers must hold a current account prior to opening a savings account (HasCurrentBefSavings), savings accounts must be positive (SavingsArePositive) and corporate customers must not hold savings accounts (CorporateHaveNoSavings).

Whilst SDs focus on static aspects, behaviour diagrams (BDs) concentrate on dynamics (or behaviour). BDs provide a map over the constituent units of a package's behaviour defined in separate diagrams. BD units of Bank (Fig. 2a) include the operations available to the environment, namely, create customer, open account, deposit, withdraw, delete account, view an account's balance, get accounts in debt, and get accounts of a customer.

In BDs, operations that modify state are represented as *contracts* (double-lined elongated hexagons); those that observe (or enquire) state as assertions (single-lined hexagons). Global operations (visible to the environment) stand alone; local operations (invisible to the environment) are placed inside the contours of their class sets. Local modifier operations can either create new class objects (constructors, symbol $\mathbb{N}$), update existing objects (symbol $\mathbb{U}$) or delete class objects (symbol $\mathbb{D}$). For instance, the global OpenAccount does all that is involved in opening actual bank accounts and the New operation inside Account, a sort of sub-operation, creates new account objects only. Modifier operations are defined in contract diagrams (CDs); operations CreateCustomer, Customer.New, AccWithdraw and Account.Withdraw of Fig. 2a are defined in CDs of Figs. 2b, 2c, 2d and 2e, respectively. Observe operations are defined in ADs; ADs of Figs. 2h and 2g define the local Account.GetBalance and the global AccGetBalance of Fig. 2a; AD of Fig. 2f defines GetAccountGivenAccNo imported in Figs. 2e and 2g.

ADs and CDs comprise a box with an identifier to the top, followed by compartments for declarations (top) and predicate (bottom) which may be further split in two. ADs have one predicate compartment as only one set of states is being referred to; CDs have two predicate compartments corresponding to the states of pre- and post-condition (to the left and right, respectively). Declaration compartments include relevant variables (either internal or of inputs and outputs), and imported assertions and contracts. Predicate compartments are made-up of graphical formulas read from top to bottom and combined using conjunction. Two

kinds of formulas are supported: logic- and set-based. Logic formulas, read from left to right, resemble their textual counter-parts. Set formulas start from some inner graphical expression and grow outwards.

AD SavingsArePositive (Fig. 1e) expresses a local invariant of Account. It has no declarations; the predicate contains a pictorial propositional logic formula made-up of individual atomic statements involving predicate edges that are combined with an implication to say that if the account's type is savings then its balance must not be negative — $aType = savings \Rightarrow balance \geq 0$.

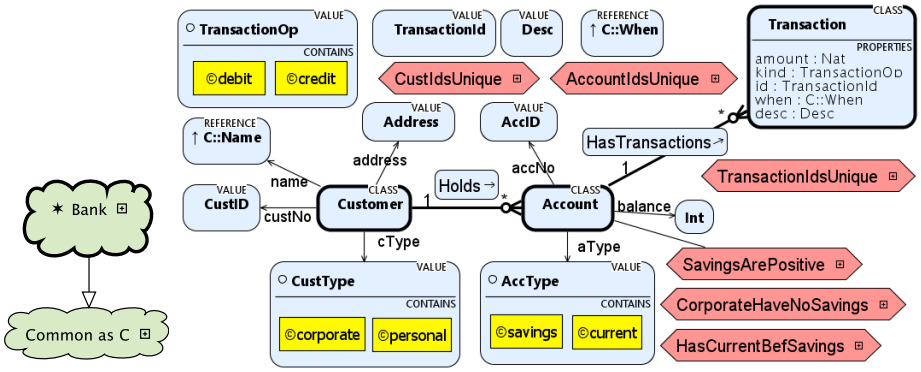
Predicate of AD CorporateHaveNoSavings (Fig. 1f) expresses a set formula. Relation Holds defined in SD of Fig. 1b, which denotes a set of pairs, is restricted to those pairs with corporate customers and savings accounts (a sort of filtering); the restricted relation is then required to be empty — hence, corporate customers must not hold savings accounts. In Fig. 1f the restrictions are performed using domain restriction (symbol $\lhd$) and range restriction ($\rhd$) using edge modifiers (represented as double arrows and denoting functions) and the outer shading indicates that the restricted set or relation must be empty. The resulting formula is: $\{o : Customer \mid o.cType = corporate\} \lhd Holds \rhd \{o : Account \mid o.aType = savings\} = \varnothing$.

AD HasCurrentBefSavings (Fig. 1d) says that the set of customers with current accounts is a subset of customers with savings accounts — hence, a customer must hold a current account prior to holding a savings account. This involves two internal set variables (included in the declarations compartment) to represent customers with current accounts (custsCurr) and customers with savings accounts (custsSav), which are defined in the predicate in a similar way: relation Holds is range-restricted using edge modifiers (symbol $\rhd$) to accounts that are either current or savings, and the actual sets are obtained from these restricted relations using the domain relational operator (symbol $\leftarrow$); finally, the bottom-most formula says, using enclosure (or insideness), that custsSav must be a subset of custsCurr. This results in the following formulas:

$$custsCurr = \mathrm{dom}(Holds \rhd \{o : Account \mid o.aType = current\})$$
$$custsSav = \mathrm{dom}(Holds \rhd \{o : Account \mid o.aType = savings\})$$
$$custsSav \subseteq custsCurr$$

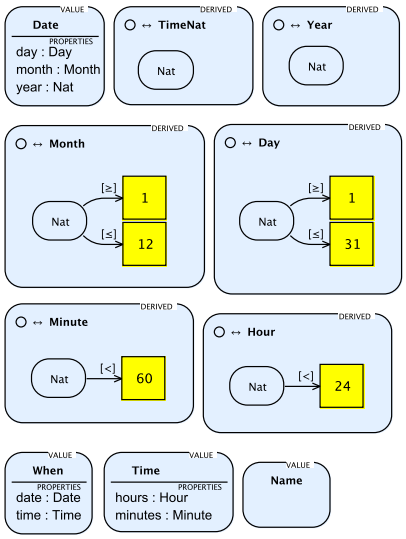
AD GetAccountGivenAccNo (Fig. 2f) fetches the Account object corresponding to aNo? into output a! — $a! \in \{o : Account \mid o.accNo = aNo?\}$. AD Account.GetBalance (Fig. 2h), which stores the account's balance in the output bal!, is imported in global AccGetBalance (Fig. 2g), which fetches the account object corresponding to the aNo? input (via imported GetAccountGivenAccNo) and calls Account.GetBalance on this object.

CD Customer.New (Fig. 2c), a constructor, says how a new Customer object (c!) should be initialised in the post-condition; using predicate edges, the after-state (represented in bold) of custNo is set non-deterministically, and those of name, addr and cType are set to the corresponding inputs. CD Account.Withdraw (Fig. 2e) declares an amount natural number input and says in the post-condition that the new account's balance (bold line around rectangle) is the old balance minus the requested amount. These two local operations are brought into the global context (of a system or package) through CreateCustomer (CD in Fig. 2b) and AccWithdraw (CD in Fig. 2d). CD of CreateCustomer declares inputs required from the environment and imports Customer.New; CD of AccWithdraw declares aNo?

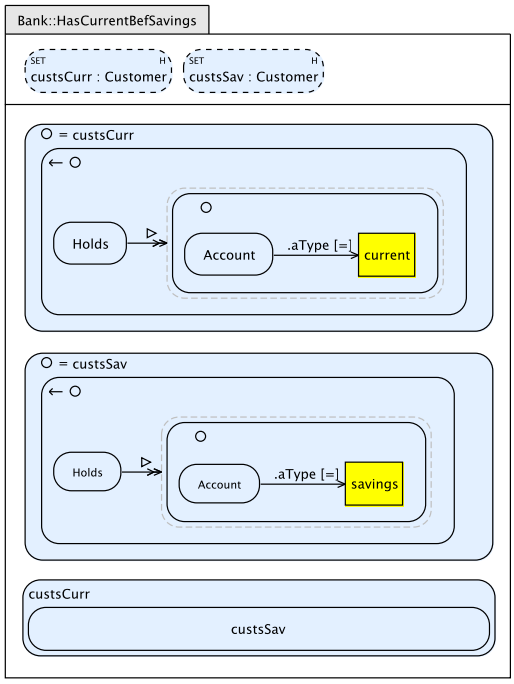


(a) PD of package Bank

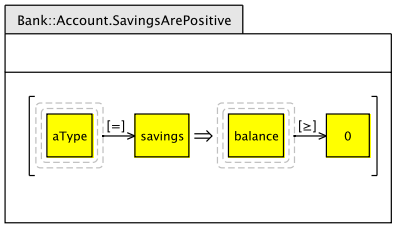
(b) SD of package Bank



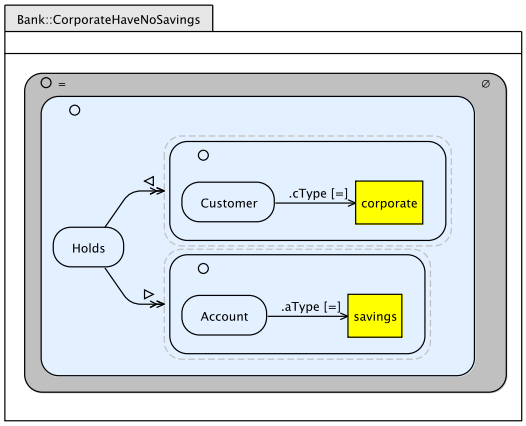
(c) SD of package of Common



(d) AD invariant HasCurrentBefSavings

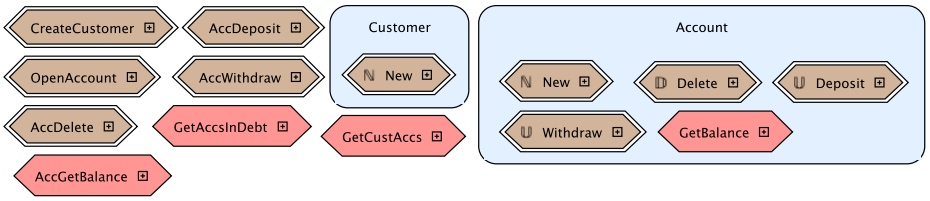


(e) AD invariant Account.SavingsArePositive

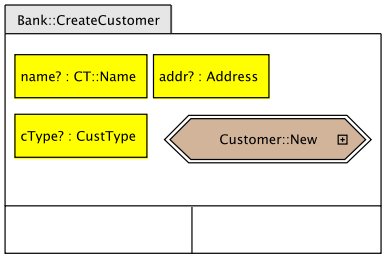


(f) AD invariant CorporateHaveNoSavings

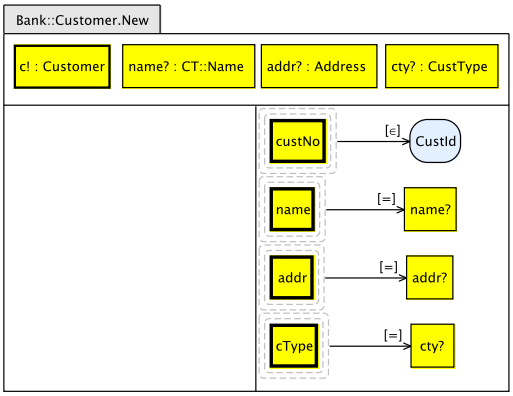
Fig. 1: Diagrams of the state space VCL model of secure simple bank (from [5]).



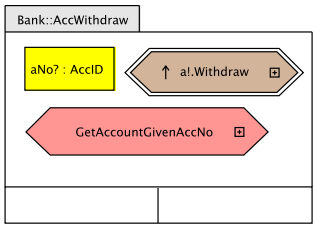
(a) Behaviour diagram



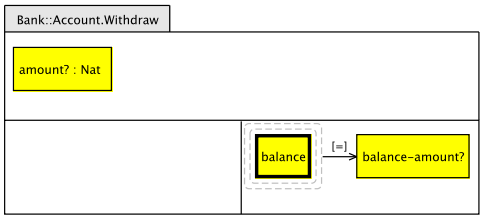
(b) Operation CreateCustomer



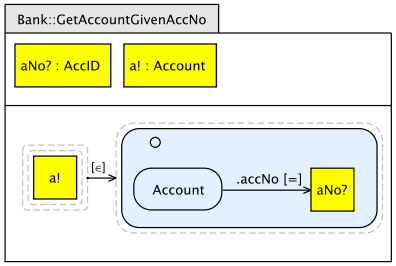
(c) Operation Customer.New



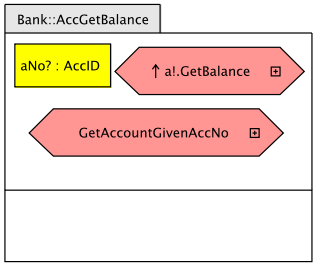
(d) Operation AccWithdraw



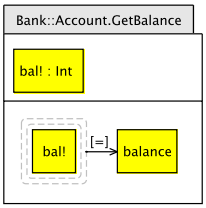
(e) Operation Account.Withdraw



(f) Operation GetAccountGivenAccNo



(g) Operation AccGetBalance



(h) Operation Account.GetBalance

Fig. 2: Diagrams of dynamic VCL model of package Bank in secure simple bank (from [5]).

input for account from which money is to be withdrawn, and imported assertion `GetAccountGivenAccNo` of (Fig. 2f).

VCL's ADs and CDs, illustrated in the VCL model excerpts of Figs. 1 and 2, are major novelties of VCL, giving VCL a strongly graphical character.

2.3 VCL and its Visual Effectiveness

VCL's design follows theories of visual notation design, namely, physics of notations (PoN) [79] and cognitive dimensions of notations (CDN) [**?**,54, 24]. The following exemplifies how these theories are applied in VCL using Figs. 1 and 2:

- VCL tries to be well-matched to meaning, following CDN's *closeness of mapping* and PoN's *semantic transparency*, by conveying the underlying mathematics. For example: VCL's round contour set construct (eg. `Customer`, `Account`, `CustId` and `CustType` in Fig. 1b, and `Holds`, `Account` and `Customer` in Fig. 1f) taps to similar shapes of mathematics (Venn or Euler circles); the shading of Venn diagrams is used in Fig. 1f to indicate that the set must be empty; the single and double lines of assertions and contracts (see Fig. 2a), respectively, refer to the fact that assertions involve a single set of states, whereas contracts involve the state-sets of pre- and post-conditions.
- VCL's graphical primitives follow PoN's principle of semiotic clarity. In the different diagrams of Figs. 1 and 2, sets are consistently rendered as rounded shapes, and values and objects (members of a set) as rectangles[2], constraints upon single states are hexagons (independently of whether they denote invariants or observe operations). Furthermore, VCL's primitives have a core meaning that varies slightly with the context, enabling users to infer the meaning of graphical expressions in different contexts, following CDN's *consistency* and PoN's *graphical economy*. The round contours of Fig. 1b do not mean exactly the same as the several round shapes of ADs and CDs of Figs. 1 and 2, but they all have set-like meanings.
- PoN's principles of *perceptual discriminability* and *visual expressiveness* can be observed in the panoply of shapes and colours used across the different VCL diagram elements illustrated in Fig. 1 — packages are green clouds, sets are rounded blue contours, assertions are red hexagons and contracts are brown hexagons, objects are yellow rectangles, shading distinguishes empty from non-empty sets, and size and brightness differentiate types of sets and edges.
- PoN's *cognitive integration* (integration of different pieces of information), and CDN's *role-expressiveness* (how pieces contribute to the whole) is intrinsic to VCL. In SDs, as illustrated in Fig. 1b, the assertions of `Bank`'s SD, defined separately in ADs, explicitly say that the AD-defined invariants constrain the SD's defined state space. BDs, on the other hand, identify all the different pieces of behaviour defined separately in ADs and CDs, as illustrated in Fig. 2a. Importing mechanism in ADs and CDs (illustrated in Figs. 2b, 2d and 2g) integrate pieces defined elsewhere to make compound definitions. Furthermore, the '+' attached to package (clouds) assertions and contracts (elongated hexagons) of SDs, BDs and PDs, provide navigation clues that a separate diagram is opened upon double-clicking. CDN's *role-expressiveness* is also manifested in SDs when

[2] UML represents both classes and objects as rectangles.

line size and brightness are used to distinguish class from values sets (class contours are thicker) to give relevance to classes which are the major abstractions of a domain and act as beacons.
- PoN's *dual coding* (text complements graphics to strengthen communication) and CDN's *secondary notation* is applied in SDs of Figs. 1b and 1c to distinguish the different kinds of sets and to reinforce the multiplicity constraints of relations; sets include a word to indicate set-kind, class sets are bold-lined and remaining sets have lines with normal thickness; relation multiplicity is conveyed both visually and textually. Figure 1f reinforces the empty set meaning through both shading and symbol $\varnothing$.
- PoN's *complexity management* (representing information without overloading the human mind) and CDN's *abstraction gradient*, are intrinsic to VCL. Statics and dynamics are clearly separated through VCL structural and behaviour diagrams; UML represents data and operations in class diagrams that tend to become severely cluttered even for medium to small models. VCL's package construct (represented as clouds, Fig. 1a) defines large modules to keep the contents of each package manageable; reference sets (symbol $\uparrow$) enable references to sets from other packages. VCL operations are constructed modularly; operation and assertions may be composed of other modules (operations or assertions); for example, in Fig. 2d, operation `AccWithdraw` is made-up of observe operations `GetAccountGivenAccNo` and local contract `Account.Withdraw`.
- VCL addresses CDN's *hard mental operations* dimension as part of its raison d'être as it tries to improve the usability of formal software design, with its inherently hard underlying mathematics, through visualisation. However, this per-se does not totally solve this problem and two principles discussed above, CDN's abstraction gradient and PoN's complexity management, are key in giving VCL an abstract and modular ethos, which helps in dealing with hard mental operations. The notion of separation, inherent to modularity, is manifested in the different compartments of ADs and CDs (Figs. 1 and 2), which ease the hardness of the task through order and focus as each compartment expresses something specific that is relevant to the whole (also an application of the cognitive integration principle). Furthermore, modellers are encouraged to come up with abstractions and express designs that are modular as VCL provides the means to break down potentially overwhelmingly complex problems into manageable and meaningful chunks. BDs, for instance, encourage abstraction and modularity by letting the modeller focus on the different pieces that make up an overall behaviour. Despite this, there are so many ways in which VCL could be improved to ease hard mental operations.

2.4 VCL's Trajectory, Founding Ideas and Suitability as Visual Notation

VCL embodies ideas of both graphical and formal modelling. A major influence is Amálio's PhD thesis [4] which proposes UML+Z [18,19], a modelling approach combining UML with the formal language Z. VCL's Z semantics [4,17,**?**] was borrowed from this work. Another inspiration is the work lead by Kelsen on diagrammatic expression of behaviour in the language EP [66,67,11].

VCL emphasises rigour, formality and modularity. Early works developed the concept with diagrams built using drawing tools [12,15,13,14]. A major influence

was the development of VCL's aspect-oriented modelling approach [15]. Once VCL had been developed as a concept, we embarked upon the construction of VCL's tool, the Visual Contract Builder (VCB)[3], which made VCL more tangible and firmly defined [10,9]. The tool paved the way to the use of VCL for coursework and student projects [72,107], brought the language to life, and made the experiment presented here possible. Empirical results on modelling tools, emerging from a spin-off survey of the experiment presented here, highlighted that both VCL and its tool were being positively received by users [9].

VCL is an experimental language embodying ideas on visual and formal modelling. It was designed to not drastically deviate from UML; it keeps the same OO foundation which has proved suitable for software design. We wanted to improve the graphics, precision and connection to mathematical modelling. VCL SDs, for instance, introduced just a few novelties with respect to class diagrams to exploit the idea of having more self-contained and closely integrated diagrams focussed on data modelling, but avoiding drastic divergences. The statecharts [57] notation was inspirational; although developed independently from UML or any of its main predecessors, it ended-up incorporated in the standard. Likewise, we hope that VCL ideas can be incorporated into such standards if they prove to be useful.

VCL's major novelty, its capacity to express predicates visually, makes it a prominent representative of graphical design languages. There are other languages with formal basis and capable of expressing predicates visually, such as augmented constraint diagrams (ACDs) [48] and Visual OCL (VOCL) [27,41]; however, VCL's most salient difference lies in its superior tool support. In a comparative study focussed on practical application [107], VCL outperformed ACD and VOCL. VCL's VCB tool outperformed the UML tool Papyrus in the comparison of tools used in the controlled experiment presented here that focussed on usability [10]. VCL is the only visual design language expressing visual predicates that tackles modelling in the large through its modelling primitives inspired by aspect-orientation [15]. In terms of usability, VCL appears to outperform both ACD and VOCL in what respects the use of colour for visual expressivity. VCL was applied to case studies proposed as challenges by research communities, such as the large car-crash crisis management system [15] and its variant the Barbados car-crash management system [6,80], and a cardiac pacemaker [73,72].

VCL is being used in student projects at undergraduate and masters level with education being a modest success. It has been used by many students to design systems and applications. Further developments of VCL could focus on: (a) VCL and its tool, aiming towards code generation, (ii) empirical experimentation, following from the results presented here, and (iii) VCL's formal foundations.

3 The Experiment's Scope

The study presented here examines VCL as a visual design notation. It seeks to know whether VCL and its capacity to express predicates visually provide any advantage over existing standard notations, in particular UML and its OCL satellite textual notation, from the perspective of modellers and end-users. The sequel dis-

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

tils the aims of the study into the experiment's objective and research questions, which are then translated into tasks and hypotheses.

3.1 Objective

The experiment's objective is as follows:

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

To accomplish this, two perspectives are considered: *modellers* and *end-users*. As said above, the experiment gauges *effectiveness*, which is meant here to be the degree to which something is successful or adequate in producing a result or accomplishing a purpose[4]. This involves evaluating performance, which means how effective are modellers or end users in accomplishing tasks.

To fulfil its objective the experiment investigates the following: (i) *modelling*, which assesses performance in building design models and perceptions emerging from doing so; (ii) *problem comprehension*, which evaluates the problem comprehension gained from modelling; (iii) *model usage*, which assesses the performance of end-users in tasks related to the usage of models, namely defect detection and model comprehension; (iv) *usefulness, ease of use, usability and overall appraisal*, which assess perceptions emerging from experiment experiences.

3.2 Research Questions

The experiment seeks answers to the following research questions (RQs):

- *RQ1: Is the performance of modellers in building software designs better with VCL than UML+OCL?*
- *RQ2: Is the comprehension of the problem accrued from modelling better with VCL than UML+OCL?*
- *RQ3: Is end-users' performance in tasks related to usage of software designs, namely defect detection and model comprehension, 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?*

3.3 Dependent Variables

The dependent variables hold measures to assess the different RQs; they are sampled according to independent variables the most determinant of which being the notation (either VCL or UML). RQ1 to RQ3 are measured both objectively and subjectively. RQ4 to RQ6 are measured subjectively. All RQs are analysed quantitatively; RQ6 is analysed qualitatively also.

[4] From definitions of effectiveness in Oxford English dictionary and Dictionary.com.

Category/RQ	Definition	Description
Completeness (Co), RQ1	$\frac{score}{maxScore}$	Aggregate proportion obtained from measuring satisfaction of each expected model piece on an ordinal scale from 4 (fully satisfied) to 0 (unsatisfied). **Variables**: CoS, CoI, CoO
Accuracy (Ac), RQ1	$\frac{score}{maxScore}$	Aggregate proportion obtained from evaluating test cases on an ordinal scale from 4 (fully satisfied) to 0 (unsatisfied). **Variables**: AcS, AcI, AcO
Perceived modelling (PM), RQ1	$\{VCL, UML, NP\}$	Notation perceived as providing better modelling performance. **Variables**: PMS, PMI, PMO
Problem comprehension (PC), RQ2	$\frac{correctAnswers}{noQuestions}$	Proportion of correct answers in PC questionnaire. **Variables**: PC
Perceived PC (PPC), RQ2	$\{VCL, UML, NP\}$	Notation perceived as providing better PC performance. **Variables**: PPC
Model comprehension (MC), RQ3	$\frac{correctAnswers}{noQuestions}$	Proportion of correct answers in MC questionnaire. **Variables**: MC
Defect detection (DD), RQ3	$\frac{foundDefects}{totalDefects}$	Proportion of identified defects in model with seeded defects (DD). **Variables**: DD
Perceived usage, RQ3	$\{VCL, UML, NP\}$	Notation perceived as providing better DD or MC performance. **Variables**: PDD, PMC
Usefulness (U) and ease of use (EoU), RQ4	$\frac{score}{maxScore}$	Calculated from responses on a Likert Scale from 1 (strongly agree) to 5 (strongly disagree). **Variables**: U, EoU
Usability (Us), RQ5	$\{VCL, UML, NP\}$	Notation perceived as better for different usability criteria. **Variables**: UsR, UsN, UsMOs UsLEC, UsLF, UsC, UsL, UsCS
Appraisal (Appr), RQ6	$\{Positive, Negative, Neutral\} \rightarrow Integer$	Subject's appraisal of VCL in comments to open questions to obtain number of positive, negative and neutral comments. **Variables**: Appr
Preferred notation (PN), RQ6	$\{VCL, UML, NP\}$	Preferred notation at state space (S), invariants (I), operations (O), overall and to use in the future. **Variables**: PNS, PNI, PNO, PN, FN

Table 1: Dependent variables. (S = state space; I = invariants; O = operations; DD = defect detection; PDD = perceived DD; PMC = perceived MC; 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.)

Table 1 summarises the dependent variables; it comprises columns for variable's category and RQ, variable definition and description. Variables' names follow a 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). For example, *CoS* is completeness of state space. We use two types of variables: (i) *proportion* (obtained quantity is divided by maximum quantity, yielding a continuous number between 0 and 1); and (ii) *nominal* or *categorical* (possible values drawn from a bounded discrete set).

The sequel gives further details on how the different RQs are assessed.

3.3.1 RQ1, Modelling

RQ1's tasks involve building a design from a given case study narrative. The design is partitioned into: state space, constraints over the state space (invariants) and behaviour as operations made-up of pre- and post-conditions (contracts).

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* (Co) measures how much is modelled. This is based on a breakdown of requirements and corresponding modelling pieces whose satisfaction is marked manually on an ordinal scale from 4 (fully satisfied) to 0 (unsatisfied).
- *Accuracy*(Ac) measures the quality of what is modelled based on a partitioning of requirements into aspects of interest exercised by test cases, which are evaluated manually on an ordinal scale from 4 (fully satisfied) to 0 (unsatisfied).

This measurement apparatus , detailed in [8], aims to make the grading objective, repeatable and unbiased. Completeness and accuracy variables of table 1 hold objective measures of modelling performance in state space (CoS and AcS), invariants (CoI and AcI) and operations (CoO and AcO) in the form of aggregate proportions (obtained score divided by maximum score). Categorical variables PMS, PMI and PMO of table 1 hold subjective measures of modelling performance in state space (S), invariants (I) and operations (O).

3.3.2 RQ2, Problem Comprehension

From a set of multiple choice questions framed in a modeller perspective, RQ2 is evaluated objectively and subjectively in variables PC (a proportion of correct answers) and PPC of table 1, respectively.

3.3.3 RQ3, Model Usage

This is evaluated with the following tasks:

- *Defect detection* (DD), related to model inspection, consists of identifying defects in a model with seeded errors.
- *Model Comprehension* (MC), or how well end-users understand given models, is assessed through multiple choice questions about a given design.

Variable *DD* of table 1 provides an objective measure as a proportion of encountered defects. Variable *MC*, focussed on the end-user perspective, measures the proportion of correct questions in the MC questionnaire.

3.3.4 RQ4, Usefulness and Ease of Use

The examined notations are assessed at the light of *perceived usefulness* (PU) and *perceived ease of use* (PEoU) [39] in the debriefing survey. PU is the degree to which someone believes that a particular system would enhance their performance. PEoU is the degree to which someone believes that a particular system would be free of effort. Both PU and PEoU are seen as important determinants for user acceptance of a technology [39]. The corresponding variables, *U* and *EoU* (table 1), hold an aggregate proportion calculated from Likert scaled statements.

3.3.5 RQ5, Usability

The debriefing survey assesses RQ5 based on relevant criteria, namely: readability, navigation, maps and overviews, live error checking, look and feel, learnability and comfort/satisfaction. Usability measures are held in the Us variables of table 1.

3.3.6 RQ6, Overall Perception

The debriefing survey enquires about overall perceptions of the examined languages based on experiment experiences. The relevant RQ6 dependent variables of table 1 are: *Appr* (an appraisal of positive, negative and neutral aspects of VCL based on the open-ended questions of the debriefing survey) and the variables that gauge the preferred notation with respect to state space (PNS), invariants (PNI), operations (PNO), overall (PN) and future usage (FN).

3.4 Hypotheses

There is one major independent variable for used notation — it has two treatments: *VCL* and *UML*. The hypotheses are formulated from independent and dependent variables (table 1). Each dependent variable has corresponding null, H_i^0 (no difference between the notations), and alternative, H_i^a (there is a difference between the notations), hypotheses. For example, completeness of state space gives $H_1^0 : CoS(VCL) = CoS(UML)$ and $H_1^a : CoS(VCL) \neq CoS(UML)$; 11 out of 31 hypotheses follow this template. Subjective dependent variables are three-valued, either *VCL*, *UML* or *NP* (no preference); the underlying hypotheses cater to this; for example, perceived modelling of state space gives $H_7^0 : PMS(VCL) = PMS(UML) = PMS(NP)$ and $H_7^a : PMS(VCL) \neq PMS(UML) \neq PMS(NP)$; 19 out of 31 hypotheses follow this pattern. Hypothesis H_{31} is formulated based on the values positive, negative and neutral; the first two are co-related to *VCL* and *UML* respectively; this gives: $H_{31}^0 : Appr(Positive) = Appr(Negative) = Appr(Neutral)$ and $H_{31}^a : Appr(Positive) \neq Appr(Negative) \neq Appr(Neutral)$.

Task 1	Task 2	Task 3	Task 4	Task 5
Modelling of state space and invariants, 30 minutes	Modelling of operations, 35 minutes	Problem comprehension, 15 minutes	Defect Detection, 25 minutes	Model Comprehension, 15 minutes

Table 2: Sequence of five tasks of an experiment session.

3.5 Recruitment

Participants were recruited from the following institutions: (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 rewarded with €50 vouchers for taking part in the experiment and were required to have completed or be in the process of completing a course on UML-based software design.

4 Experiment Design

The following explains the major elements of the experiment's design.

4.1 Case Studies

The experiment's case studies, detailed in [8], are as follows:

- *University Library (UL).* A university library system enables members to borrow and return books, renew borrowings and recall books unavailable for loan.
- *Flight Booking (FB).* A system to manage flight bookings of different airlines.

4.2 Experimental Tasks and Time Allocation

The experiment's tasks feed the dependent variables of table 1. Each session required participants to work as both modellers and end-users, using either VCL or UML+OCL in either one of the two case studies. A session lasted two hours and comprised the following sequence of tasks (summarised in table 2):

1. *Modelling of state space and invariants* (variables CoS, CoI, AcS and AcI, table 1) required completing a data model of either case study within 30 minutes, using either UML class diagrams (CDs) and OCL, or VCL structural and assertion diagrams.
2. *Modelling of operations* (variables CoO and AcO, table 1) involved modelling two operations in a given data model within 35 minutes, using either OCL and UML CDs, or VCL behaviour, contract and assertion diagrams.
3. *Problem comprehension* (variable PC, table 1) involved completing a questionnaire focussed on the modelled problem within 15 minutes.

	Day 1	**Day 2**	**Day 3**	**Day 4**
Group A	VCL, UL	UML+OCL, UL	UML+OCL, FB	VCL, FB
Group B	UML+OCL, UL	VCL, UL	VCL, FB	UML+OCL, FB

Table 3: Experiment's scheduling, highlighting case study and language used by each group on each round. (UL = University Library; FB = Flight Booking.)

4. *Defect Detection* (variable *DD*, table 1), about finding seeded defects in complete models (comprising both static and dynamic parts) within 25 minutes, required browsing through either UML CDs and OCL, or VCL structural, assertion, behaviour and contract diagrams.
5. *Model Comprehension* (variable *MC*, table 1), about completing a questionnaire on a given complete model within 15 minutes, involved browsing through UML CDs and OCL, and a VCL complete model.

Participants worked on each of the two systems, using VCL or UML+OCL. They were prevented from collaborating; the work was monitored as tasks were performed in a classroom. Although aware of the experiment's overall goal, participants were unaware of experimental hypotheses or dependent variables.

4.3 Design Type and Scheduling

The experiment follows a *crossover* or *within-subjects* design — all participants are subject to the different treatments. It involves one main factor (or treatment), the design notation (either VCL or UML+OCL), and a secondary case study factor (either UL or FB), yielding four treatment combinations of a 2^2 factorial design. The rationale is: (a) maximise number of data points to increase statistical power, (b) remove or mitigate bias emerging from differences in case-study complexity or individual ability, (c) at the cost of possible carry-over effects [55].

Participants were split into two groups. The group-assignment was mainly based on availability due to the experiment's voluntary nature. However, an effort was made to scatter the high ability individuals evenly among groups based on the results of an ability questionnaire; this is elaborated further in section 7.

Table 3 outlines the experiment's four rounds. In each round, each group is given a different treatment. All participants were subject to the four treatment combinations.

4.4 Modelling Tools

The experiment relied on modelling tools. This enhances language usability, contributes to the quality of resulting models and ensures a connection with modern-day reality, which relies heavily on software tools. The experiment uses two Eclipse-based tools: Visual Contract Builder (VCB) [9,10][5] and Papyrus[6]. VCB is the only VCL tool. Papyrus was chosen because it is Eclipse-based, which ensures a certain degree of similarity.

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

[6] `http://www.eclipse.org/papyrus/`

	Session 1	Session 2	Session 3
FCUL	VCL (2 hours)	UML+OCL (2 hours)	–
FCT/UNL	VCL (2 hours)	UML+OCL (2 hours)	1 hour VCL followed by 1 hour UML+OCL (optional)
U. Luxembourg	VCL (2 hours)	UML+OCL (2 hours)	1 hour VCL followed by 1 hour UML+OCL (optional)
U. York	UML+OCL (3 hours)	VCL (3 hours)	–

Table 4: Administered training in the different experiment replications.

4.5 Training

The training covered the examined notations, focussed on the more challenging tasks of modelling of state space, invariants and operations, and relied on the participant's university training on UML-based design. It consisted of a live and assisted modelling of a system from a requirements description using the notations under study and their supporting tools mimicking the experiment's modelling tasks. No training was given on the experiment's model usage tasks (problem comprehension, defect detection and model comprehension) due to time reasons — if participants were able to model, then they would be able to undertake the easier model usage tasks.

The training lasted 4 to 6 hours with two mandatory hours per notation. Table 4 summarises the training provided at the different experiment replications. Each participant had at least 4 hours of training (2 VCL + 2 UML) with variations across the replications. At FCUL no extra training was given. Some FCT/UNL and Luxembourg participants were given one hour extra of training. York participants received six hours of training (3 VCL + 3 UML).

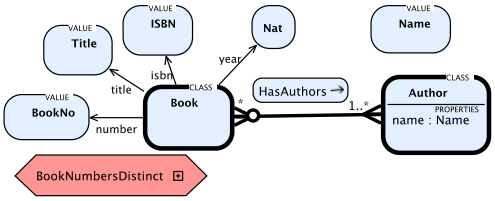
5 Instrumentation

The experiment's artefacts comprise: (a) case study narratives; (b) sample case study models for tasks of modelling, defect detection and model comprehension; (c) problem and model comprehension questionnaires; (d) ability questionnaire; and (e) debriefing survey. All materials are given in full in [8].

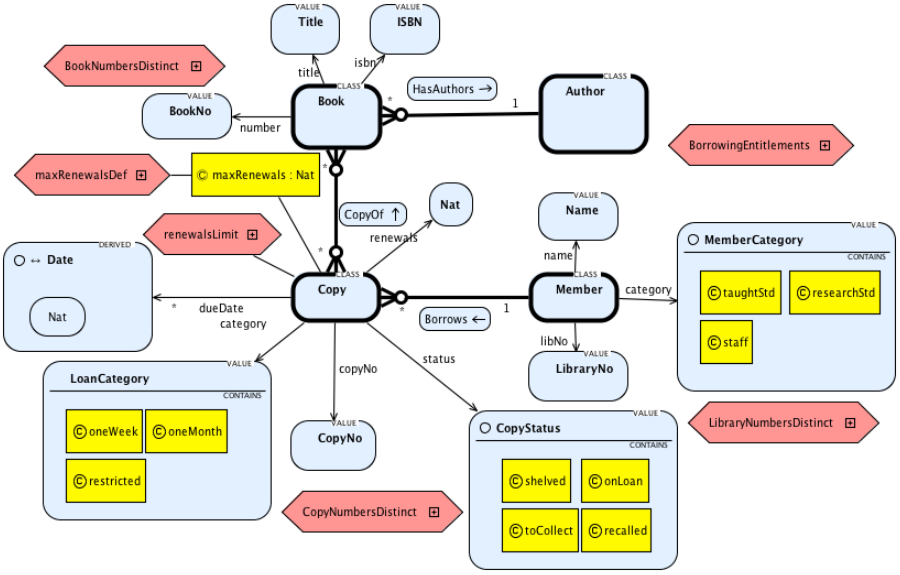
Each seeded defect and comprehension question was selected according to a number of criteria: it had to cover different aspects or parts of the system to the largest extent possible; it should neither be trivial nor overly difficult to answer or find. The questions had to be relevant and genuine, ideally questions that a software engineer could ask about the system. Standard techniques for phrasing subjective questions and designing surveys were followed [85].

5.1 Case Study Narratives and Sample Models

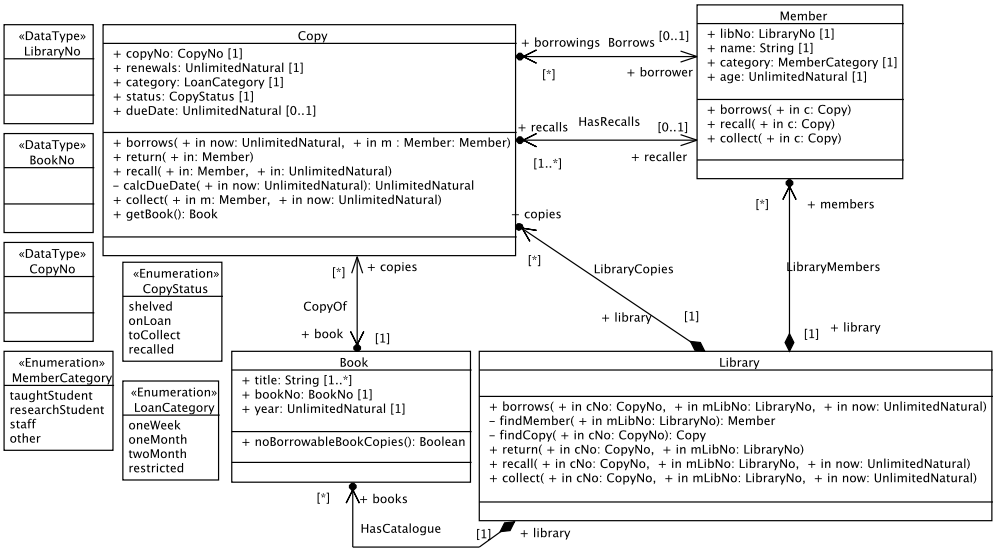
All tasks of an experiment session use either one of the two case studies, UL or FB (section 4.1), exemplars of systems used in the teaching of software design.



(a) VCL structural diagram of initial model



(b) VCL structural diagram of model with defects



(c) UML class diagram of model with defects

Fig. 3: Structural diagrams of sample models of university library.

	University Library		Flight Booking	
	VCL	**UML+OCL**	**VCL**	**UML+OCL**
State space	8	7	13	13
Invariants	7	5	7	6
Operations	21	24	19	20
Total	36	36	39	39
Goodness of fit p-value	.51		.9	

Table 5: Frequency distribution of seeded errors in models with defects. This considers both case studies; distributions are given with respect to modelling aspects (state space, invariants and operations).

At the start of a session, participants were given a requirements narrative (given in [8]). For the task on modelling state space and invariants, participants had to complete a given incomplete model (called initial model) — fig. 3a gives UL's initial VCL model. For the modelling of operations, participants had to model two operations on another given model (intermediate model with a complete state space and incomplete dynamics).

The given models aimed to get the most out of the short modelling tasks. They acted as ice-breakers, countering against the blank page effect, and contributing to the task's fluidity and engagement to provide meaningful experiment data.

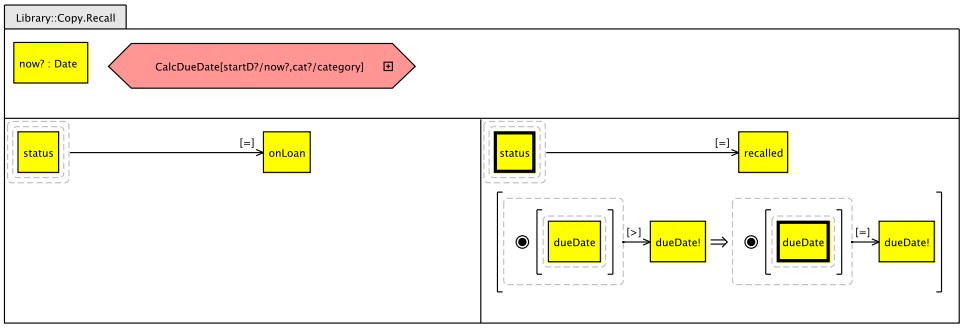
For defect detection (DD), participants were given models with seeded defects, which they had to identify by completing an online form. Figure 3b gives the SD of UL's DD VCL model and Fig. 3c gives the UML counterpart; all faulty models are given in full in [8]. Defects were seeded in the solution models of both case studies; for example, one error in Fig. 3b is that authors do not have names, whereas in Fig. 3c class `Author` is missing altogether. Table 5 gives the frequency distribution of seeded defects across the different modelling aspects being analysed (state space, invariants and operations). A χ^2 goodness of fit test confirmed that the distributions of notations and case study are evenly distributed (see p-values of goodness of fit in table 5) to avoid bias — no significant differences were found.

For model comprehension, subjects were given a complete case study model. Two operations of these models for the UL case study are given in Fig. 4.

5.2 Comprehension Questionnaires

After modelling, participants had to complete a questionnaire with 12 multiple-choice questions having one correct answer, to assess comprehension of modelled problem. A sample question is given in Fig. 5.

Model comprehension questionnaires were accompanied by a complete model of the system. They contained 12 multiple-choice questions having a unique correct answer. A sample question of a model comprehension questionnaire for the university library case study is given in Fig. 6. Whilst the problem comprehension questionnaire evaluates someone's understanding of the requirements of the modelled problem, the model comprehension questionnaire was focussed on the understanding of what is modelled and what a model actually says. Note how



(a) A VCL contract diagram for the local operation Copy.Recall

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

(b) OCL pre- and post-conditions for the local operation Copy.Recall

Fig. 4: Local operation Copy.recall of university library expressed in the notations under study.

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

Fig. 5: A sample question of a problem comprehension questionnaire for the flight booking case study.

diagrams of Fig. 4 give the answer to question of Fig. 6: when a Copy is recalled, its status changes from onloan to recalled.

5.3 Ability Questionnaire

The ability questionnaire assesses the capabilities of participants prior to the experiment. It questioned participants on whether they had completed or were in the process of completing a higher education degree in computer science, and what was their exposure to computer programming, discrete mathematics, visual modelling (using either object-oriented or structured methods), and formal modelling. Figure 7 gives two sample questions of this questionnaire.

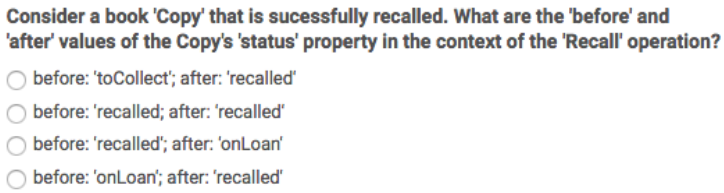


Fig. 6: A sample question of a model comprehension questionnaire for the university library case study.

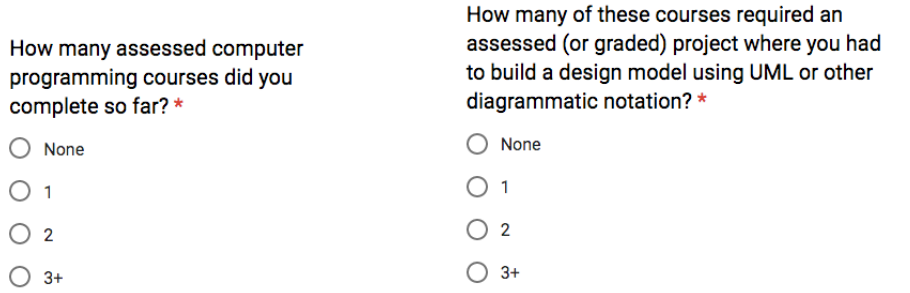


Fig. 7: Two questions of the ability questionnaire.

5.4 Debriefing Survey

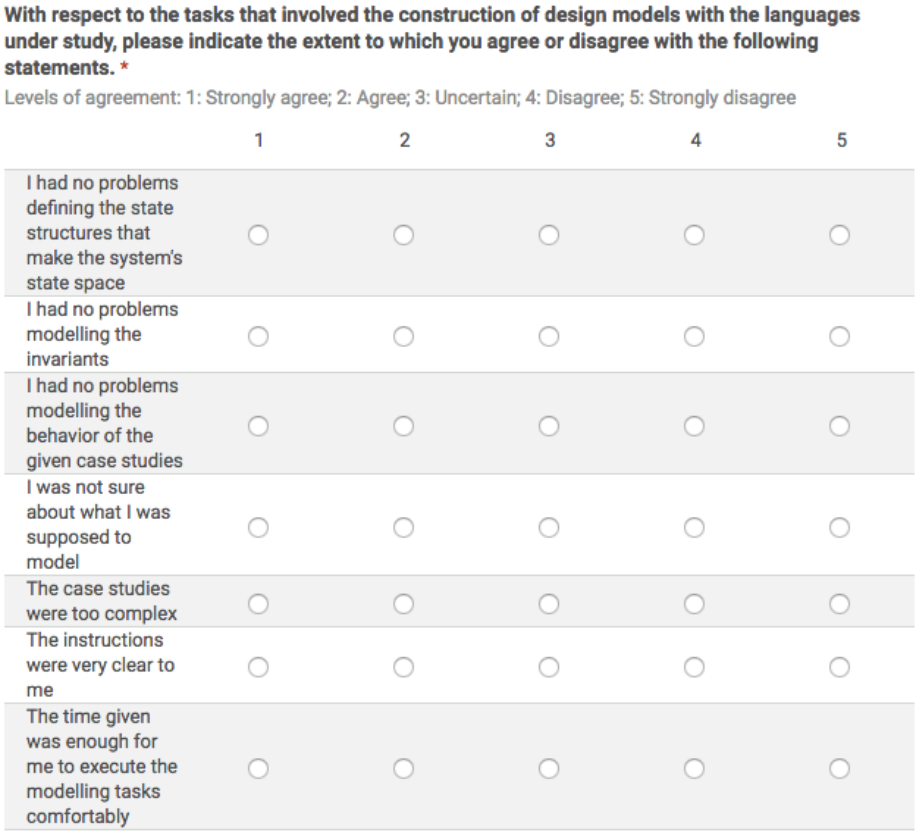
The debriefing survey gauges perceptions resulting from individual experiment experiences. This includes perceived performance in tasks of modelling, problem comprehension, defect detection and model comprehension, as well as perceived usefulness, ease of use, usability, preferred notation, and positives and negatives of the two notations under study. In addition, the debriefing survey provides supplementary information meant to support and explain the quantitative results by providing qualitative insight. A sample question is given in Fig. 8.

6 Statistical Analysis

The quantitative analysis relies on null-hypothesis significance testing (NHST), effect sizes (ESs) and confidence intervals (CIs). ESs are quantitative estimates of the magnitude of some effect of interest, often the size of a difference. CIs give the precision of point estimates or measurements, providing a range of plausible values within which we can have a degree of confidence that the estimate is not due to chance. The analysis was conducted using the R statistical software [91].

6.1 Means, Proportions and their CIs

The analysis emphasises measures of central tendency. For continuous variables, we calculate means (point estimates) whose precision is assessed through 95% CIs



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 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	○	○	○	○	○

Fig. 8: A sample question from the debriefing questionnaire.

calculated using the following formula [36]:

$$95\%CI = M \pm t_{.95}(N-1) \times SE,\ SE = \frac{SD}{\sqrt{N}}$$

Above: M is the sample's mean, SE the standard error, SD the standard deviation, and N the size of the sample; $t_{.95}(N-1)$ is critical value of the t distribution with $N-1$ degrees of freedom and corresponding to the 95% range[7].

Categorical variables are studied with proportions, derived from frequency distributions, which, like means, are estimates whose precision can be assessed with CIs. For proportion CIs, we use the more robust approach of Newcombe et al [81, 82], as recommended by Cumming [36] as it provides good approximations even

[7] The classical formula uses the value 1.96, the Z score of the 95% range in the normal distribution. However, the t approach is more robust as it handles small sample sizes (less than 30), the case of many CIs calculated here.

when N is small and P is close or equal to 0 or 1; this gives the formula[8]:

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

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

6.2 Hypothesis Testing

For a dependent variable V (table 1), null and alternative hypotheses are formulated using either continuous or categorical templates:

$$H_i^0 : V(VCL) = V(UML) \qquad H_i^a : V(VCL) \neq V(UML)$$
$$H_i^0 : V(VCL) = V(UML) = V(NP) \quad H_i^a : V(VCL) \neq V(UML) \neq V(NP)$$

Hypotheses are tested by estimating probabilities called p-values. Given data D, and some null hypothesis H_0, a p-value is a conditional probability estimate, $P(D \mid H_0)$ [34] — the likelihood of the given observations assuming that the null hypothesis is true. Rejecting H_0 means that it is unlikely because our observations deem $P(D \mid H_0)$ to be unlikely — hence, we accept the alternative hypothesis. Null hypotheses are rejected based on three levels of statistical significance, depending on whether the p-value is below $\alpha = .05$ (*), $\alpha = .01$ (**) or $\alpha = .001$ (***).

Continuous hypotheses are tested using the non-parametric Wilcoxon test (hereafter referred to as W), a robust test as it does not assume that the sample population is normally distributed. Categorical variables are tested using the χ^2 test. Calculated p-values are two-tailed.

To counter against the problems of multiple testing and to increase the reliability of NHST, the analysis uses the false discovery rate [22] and the method of Benjamini and Yekutieli (BY) [23], a more relaxed alternative to the family-wise error rate and the conservative procedure of Bonferroni [62], to calculate adjusted p-values. Null hypotheses are rejected based on adjusted p-values only.

6.3 Effect Sizes (ESs)

We use as ES measurement the raw (or unstandardised) mean difference. We consider two cases: paired and unpaired data.

Given the experiment's within-subjects design, most of our mean difference calculations fall under the paired case. The formulas are as follows [56, 87, 50, 36]:

$$PD = V_{VCL} - V_{UML},\ SE_{PD} = \frac{SD_{PD}}{\sqrt{N}},\ 95\%CI = M_{PD} \pm t_{.95}(N - 1) \times SE_{PD}$$

Above: $M_{PD} = M_{VCL} - M_{UML}$; SD_{PD} = standard deviation of paired differences.

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

The formula for the unpaired case is as follows [36]:

$$SE_{UD} = \sqrt{\frac{(N_A - 1)SD_A^2 + (N_B - 1)SD_B^2}{N_A + N_B - 2}}$$
$$95\%CI = M_{UD} \pm t_{.95}(N_A + N_B - 2) \times SE_{UD}$$

Above, we assume groups A and B, and $M_{UD} = M_B - M_A$.

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 cater for uniformity, we use the ES Cohen's d [33], which is appropriate for continuous variables and means. To estimate this, we use the standard deviation average (or pooled standard deviation) as the standardiser (denominator) [36]:

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

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 [36]. CIs for this ES are calculated using approaches based on non-central t distributions [38,65].

For categorical data, we use the ES measurement Cohen's h [33], based on the arcsine transformation, which is appropriate for differences between proportions. The formula for h and the SE to calculate CIs (from [33]) is:

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

6.4 Statistical Graphs

The paper depicts its results using the following graph types:

- *Histograms* (example in Fig. 9c), typically used to depict frequency distributions, use the area of the bars to represent proportions.
- *Plots of point estimates and confidence intervals* (example in Fig. 9e) depict point estimates (e.g. means) as dots and 95% confidence intervals (CIs) as error bars to convey the uncertainty of the sampled point estimates. They display the different samples in the abscissa and the units of the analysed data in the ordinate (Fig. 9e uses subject's proficiency scores).
- *Forest plots* (example in Fig. 12a) display point estimates and their corresponding 95% CIs. Point estimates are represented as circles and CIs as error bars; the abscissa conveys the scale of the pictured result.

6.5 Symbols

The sequel adopts the following symbols to convey statistical significance and ES:

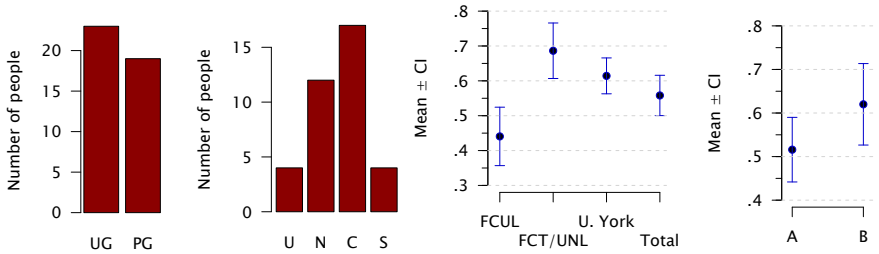
- For statistical significance, given a p-value p we have: ns = not significant ($p \geq .05$); * = $p < .05$; ** = $p < .01$, *** = $p < .001$.
- For ESs, we use symbols to denote the attained magnitude level. Given a value *es* we have: ø = null ($|\ es\ | \leq .05$); • = small ($.05 < |\ es\ | \leq .2$); •+= small to medium ($.2 < |\ es\ | < .4$); ••= medium ($.4 \leq |\ es\ | \leq .6$); ••+= medium to large ($.6 < |\ es\ | < .8$); •••= large ($|\ es\ | \geq .8$)

	M	S	U	N	C	S
FCUL	.44[.36, .52]	.16	3	9	4	0
FCT/UNL	.69[.61, .77]	.12	0	1	7	3
UL	.58[0, 1]	.25	1	1	0	1
U. York	.61[.56, .67]	.06	0	1	6	0
Total	.56[.5, .62]	.17	4	12	17	4

(a) Proficiency of the experiment's participants across experiment venues

Group	N	M	S
A	22	.52[.44, .59]	.17
B	15	.62[.53, .71]	.17
p		$.06_{ns}$	
$B - A$		.1[−.24, .44]	
d		.62[−.05, .99]●●+	

(b) Experimental groups, aggregated proficiencies and differences



(c) Subject's degree levels (d) Subject's proficiency levels (e) Means and CIs of proficiencies across venues (f) Means and CIs of proficiencies across experimental groups.

Fig. 9: A characterisation of the experiment's participants based on responses to the ability questionnaire. (UL = University of Luxembourg; U = unfit; N = Novice; C = Competent; S = Skilled; UG = Undergraduate; PG = Postgraduate.)

7 Participants

Figure 9 characterises the experiment's participants using data gathered from the ability questionnaire. Participants had diverse levels of education and training, ranging from bachelor students taking a course on object-oriented design for the first time to students undertaking their PhD studies in topics related to software engineering. The proficiency scores cater to the subjects': higher education in computer science, ability in computer programming, and exposure to discrete mathematics, OO modelling and formal modelling. Each criterion was evaluated on a scale from 0 (no competence or exposure) to 4 (high degree of competence or exposure to a subject); the final score being a proportion (obtained score divided by maximum score). From this score, participants were classified as *unfit* (between 0 and .4), *novice* (between .4 and .6), *competent* (between .6 and .8) and *skilled* (between .8 and 1). Table 9a provides the mean proficiency scores and 95%CIs of each venue and in total, and the frequency distributions across proficiency categories, both are pictured in Figs. 9d and 9e, respectively. In general, participants had the required level of training and education. All participants were, or in the course of being, university-trained in UML-based software design; 23 under- and 19 post-graduate students took part in the experiment (Fig. 9c).

Table 9b and the plot of Fig. 9f characterise the proficiencies of the two experimental groups. We can see that the proficiency of group B is non-significantly higher than A's — W p-value = .06 (ns), M_{B-A} = .1[−.24, .44], medium to large effect (d = .62[−.05, .99], ●●+)

8 Results

We start with an overview of the experiment's results (section 8.1), followed by an account of the results for: modelling (section 8.2), comprehension and model usage (section 8.3), usefulness and ease of use (section 8.4), usability (section 8.5) and overall perception (section 8.6). Since the experiment uses a crossover design we look into learning effects (section 8.7).

8.1 Results: a Bird's Eye View

Table 6 summarises the outcomes of hypotheses testing. First column (*Hyp*) gives the dichotomous outcome: either null (no difference) or alternative (a significant difference in favour of either approach) hypothesis — highlighted using shading (green in colour version). Columns $VCL_{M/P}$ and $UML_{M/P}$ give either a mean (M) or a proportion (P) of VCL and UML, respectively. Column $VCL - UML$ gives difference between either means or proportions. Column *p/q* gives NHST probabilities (p-values) with appraisals of significance (as per section 6.5); raw p-values (first value), denoted as p, are calculated using either Wilcoxon (W) or χ^2 tests; if p is significant, we obtain the adjusted p-value, denoted as q, using the false discovery rate [22] and the method of Benjamini and Yekutieli (BY) [23] to counter against multiple testing issues. Finally, column *es* (effect size) provides measures of magnitude, using either Cohen's d or h and levels of magnitude (as per section 6.5). Null hypotheses are rejected based on the adjusted p-value q.

The experiment's results are given in full in [8] with an accompanying detailed analysis. The next sections present an abridged analysis with relevant results.

8.2 Modelling

Modelling is evaluated objectively based on completeness (how much is modelled) and accuracy (quality of what is modelled), supplemented with subjective measures. We present the data using plots of means and CIs, focusing on the results for overall, each case study, and each experiment venue.

8.2.1 Completeness

The completeness results, corresponding to hypotheses $H_{1..3}$ of table 6, are portrayed in Fig. 10. They are as follows:

- In state space (Fig. 10a), VCL ($M = .64$, 95% CI $[.6, .69]$, $sd = .22$) is very close to UML+OCL ($M = .67$, CI $[.63, .71]$, sd=.19) — the mean of differences (MD) is $-.03$ (CI $[-.07, .02]$). Cohen's d ES of $-.13$ (CI $[-.35, .08]$, •) indicates a small effect.
- In invariants (Fig. 10c), VCL ($M = .1$, CI$[.07, .13]$, sd $= .14$) is close to UML+OCL (M=.13, CI$[.09, .18]$, sd $= .22$) — $MD = -.03$ (CI$[-.08, .01]$), small effect ($d = -.18$, CI$[-.38, .05]$, •).
- In Operations (Fig. 10e), VCL's advantage ($M = .16$, CI $[.13, .19]$, $sd = .14$) to UML+OCL ($M = .11$, CI $[.09, .13]$, $sd = .1$) is significant — $MD = .05[.03, .08]$, W $p = 3 \times 10^{-6}$(***), BY $q = 7 \times 10^{-5}$ (***), $d = .45[.26, .71]$ (••).

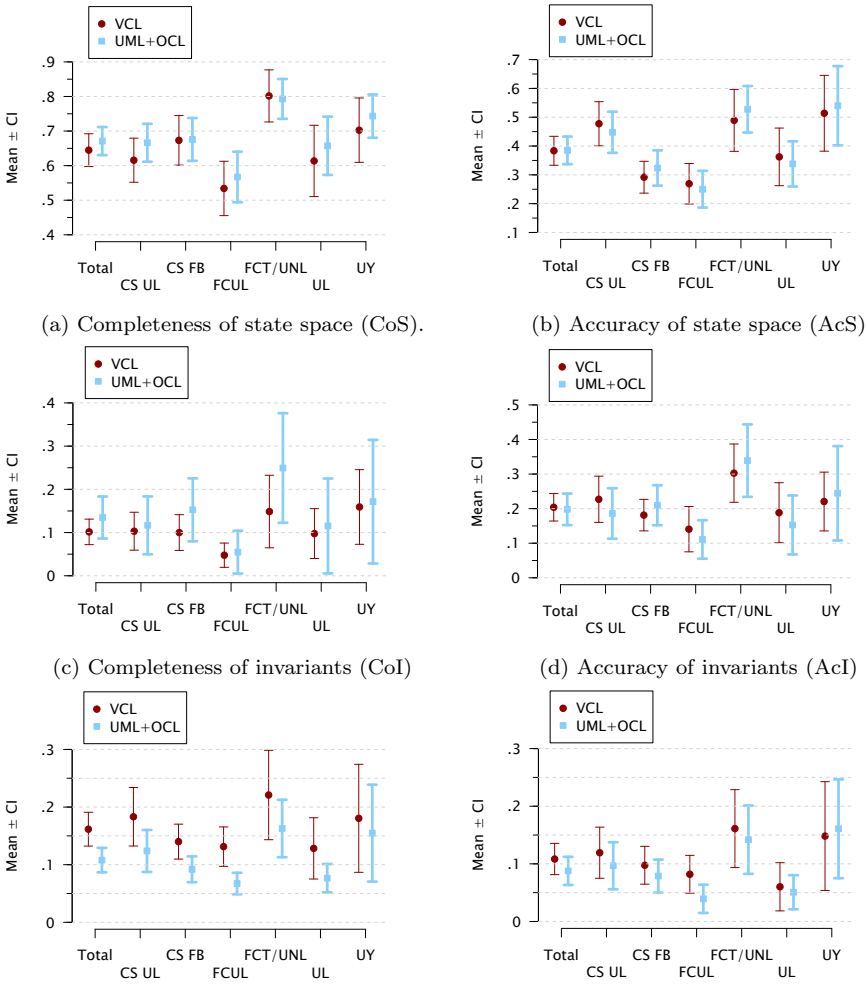
Hyp	$VCL_{M/P}$	$UML_{M/P}$	VCL - UML	p/q	$es_{(d/h)}$
Modelling (objective, RQ1)					
$H_1^0 : CoS$	$.64_M[.6,.69]$	$.67_M[.63,.71]$	$-.03[-.07,.02]$	$.23_{ns}$	$-.13^{\bullet}_{d}[-.35,.08]$
$H_2^0 : CoI$	$.1_M[.07,.13]$	$.13_M[.09,.18]$	$-.03[-.08,.01]$	$.32_{ns}$	$-.18^{\bullet}_{d}[-.38,.05]$
$H_3^a : CoO$	$.16_M[.13,.19]$	$.11_M[.09,.13]$	$.05[.03,.08]$	$3\times10^{-6}_{***}/9\times10^{-5}_{***}$	$.45^{\bullet\bullet}_{d}[.26,.71]$
$H_4^0 : AcS$	$.38_M[.33,.43]$	$.38_M[.34,.43]$	$-.001[-.05,.05]$	$.83_{ns}$	$-.01^{\emptyset}_{d}[-.22,.21]$
$H_5^0 : AcI$	$.2_M[.16,.24]$	$.2_M[.15,.24]$	$.01[-.04,.05]$	$.74_{ns}$	$.03^{\emptyset}_{d}[-.19,.24]$
$H_6^0 : AcO$	$.11_M[.08,.14]$	$.09_M[.06,.11]$	$.02[-.004,.04]$	$.07_{ns}$	$.17^{\bullet}_{d}[-.03,.4]$
Perceived modelling (subjective, RQ1)					
$H_7^0 : PMS$	$.51_P[.37,.65]$	$.35_P[.22,.5]$	$.16[-.11,.41]$	$.011_*/.088_{ns}$	$.33^{\bullet+}_{h}[.03,.63]$
$H_8^0 : PMI$	$.53_P[.39,.67]$	$.26_P[.15,.4]$	$.28[.02,.5]$	$.018_*/.13_{ns}$	$.58^{\bullet\bullet}_{h}[.28,.88]$
$H_9^a : PMO$	$.6_P[.46,.74]$	$.28_P[.17,.43]$	$.33[.05,.55]$	$.00034_{***}/.0061_{**}$	$.67^{\bullet\bullet+}_{h}[.37,.97]$
Comprehension and model usage (objective, RQ2 and RQ3)					
$H_{10}^0 : PC$	$.65_M[.61,.69]$	$.64_M[.6,.68]$	$.01[-.04,.05]$	$.66_{ns}$	$.03^{\emptyset}_{d}[-.18,.24]$
$H_{11}^a : DD$	$.27_M[.24,.3]$	$.19_M[.17,.21]$	$.08[.05,.1]$	$5\times10^{-7}_{***}/3\times10^{-5}_{***}$	$.67^{\bullet\bullet+}_{d}[.41,.87]$
$H_{12}^a : MC$	$.67_M[.64,.71]$	$.61_M[.58,.65]$	$.06[.02,.1]$	$.0025_{**}/.03_*$	$.37^{\bullet+}_{d}[.09,.52]$
Perceived comprehension and model usage (subjective, RQ2 and RQ3)					
$H_{13}^0 : PPC$	$.47_P[.33,.61]$	$.12_P[.05,.24]$	$.35[.13,.53]$	$.0098_{**}/.081_{ns}$	$.81^{\bullet\bullet\bullet}_{h}[.51,1.1]$
$H_{14}^0 : PDD$	$.56_P[.41,.7]$	$.23_P[.13,.38]$	$.33[.06,.54]$	$.0074_{**}/.066_{ns}$	$.68^{\bullet\bullet+}_{h}[.38,.98]$
$H_{15}^0 : PMC$	$.44_P[.3,.59]$	$.14_P[.07,.27]$	$.3[.08,.49]$	$.026_*/.17_{ns}$	$.69^{\bullet\bullet+}_{h}[.39,.99]$
Usefulness and ease of use (subjective, RQ4)					
$H_{16}^0 : U$	$.68_M[.64,.73]$	$.66_M[.61,.7]$	$.03[-.04,.09]$	$.19_{ns}$	$.18^{\bullet}_{d}[-.18,.42]$
$H_{17}^a : EoU$	$.58_M[.55,.62]$	$.49_M[.45,.54]$	$.09[.04,.15]$	$.0026_{**}/.03_*$	$.7^{\bullet\bullet+}_{d}[.21,.85]$
Usability (subjective, RQ5)					
$H_{18}^0 : UsR$	$.53_P[.39,.67]$	$.33_P[.2,.47]$	$.21[-.07,.45]$	$.0064_{**}/.066_{ns}$	$.43^{\bullet\bullet}_{h}[.13,.73]$
$H_{19}^a : UsN$	$.69_P[.54,.81]$	$.14_P[.07,.28]$	$.55[.29,.72]$	$6\times10^{-6}_{***}/.00014_{***}$	$1.19^{\bullet\bullet\bullet}_{h}[.88,1.49]$
$H_{20}^0 : UsMOs$	$.56_P[.41,.7]$	$.21_P[.11,.35]$	$.35[.09,.56]$	$.0074_{**}/.066_{ns}$	$.74^{\bullet\bullet+}_{h}[.44,1.04]$
$H_{21}^a : UsLEC$	$.51_P[.37,.65]$	$.09_P[.04,.22]$	$.42[.2,.59]$	$.0024_{**}/.03_*$	$.97^{\bullet\bullet\bullet}_{h}[.68,1.27]$
$H_{22}^a : UsLF$	$.7_P[.55,.81]$	$.16_P[.08,.3]$	$.53[.27,.72]$	$3\times10^{-6}_{***}/9\times10^{-5}_{***}$	$1.15^{\bullet\bullet\bullet}_{h}[.85,1.45]$
$H_{23}^0 : UsC$	$.37_P[.24,.52]$	$.27_P[.16,.42]$	$.1[-.14,.32]$	$.68_{ns}$	$.21^{\bullet+}_{h}[-.1,.52]$
$H_{24}^0 : UsL$	$.46_P[.3,.64]$	$.32_P[.18,.51]$	$.14[-.18,.43]$	$.27_{ns}$	$.29^{\bullet+}_{h}[-.08,.66]$
$H_{25}^0 : UsCS$	$.54_P[.36,.7]$	$.25_P[.13,.43]$	$.29[-.04,.55]$	$.074_{ns}$	$.6^{\bullet\bullet}_{h}[.22,.97]$
Overall Perception (subjective, RQ6)					
$H_{26}^0 : PNS$	$.42_P[.28,.57]$	$.44_P[.3,.59]$	$-.02[-.29,.24]$	$.026_*/.17_{ns}$	$-.05^{\emptyset}_{h}[-.35,.25]$
$H_{27}^a : PNI$	$.65_P[.5,.78]$	$.19_P[.1,.33]$	$.47[.2,.66]$	$6\times10^{-5}_{***}/.0012_{**}$	$.99^{\bullet\bullet\bullet}_{h}[.69,1.28]$
$H_{28}^a : PNO$	$.58_P[.43,.72]$	$.21_P[.11,.35]$	$.37[.11,.58]$	$.0026_{**}/.03_*$	$.78^{\bullet\bullet+}_{h}[.49,1.08]$
$H_{29}^0 : PN$	$.49_P[.35,.63]$	$.28_P[.17,.43]$	$.21[-.05,.44]$	$.091_{ns}$	$.43^{\bullet\bullet}_{h}[.14,.73]$
$H_{30}^0 : FN$	$.33_P[.2,.47]$	$.33_P[.2,.47]$	$0[-.23,.23]$	$.98_{ns}$	$0^{\emptyset}_{h}[-.3,.3]$
$H_{31}^a : Appr$	$.56_P[.51,.61]$	$.31_P[.27,.36]$	$.24[.15,.33]$	$4\times10^{-24}_{***}/4\times10^{-22}_{***}$	$.5^{\bullet\bullet}_{h}[.4,.6]$

Table 6: Results of hypotheses testing. Confirmed alternative hypotheses are highlighted (in shaded or green in colour version).

8.2.2 Accuracy

The accuracy results, corresponding to hypotheses $H_{4..6}$ of table 6, are portrayed in Fig. 10. They are as follows:

– In state space (Fig. 10b), VCL ($M = .38[.33,.43]$, $sd = .23$) is quasi-equal to UML ($M = .38[.34,.43]$, $sd = .22$) — $MD = 0[-.05,.05]$, $d = -.01[-.22,.21]$ (ø). Likewise for invariants (Fig. 10d) — VCL ($M = .2[.16,.24]$, $sd = .18$), UML ($M = .2[.15,.24]$, $sd = .21$), $MD = .01[-.04,.05]$, $d = .03[-.19,.24]$ (ø).



(a) Completeness of state space (CoS).

(b) Accuracy of state space (AcS)

(c) Completeness of invariants (CoI)

(d) Accuracy of invariants (AcI)

(e) Completeness of operations (CoO)

(f) Accuracy of operations (AcO)

Fig. 10: Model completeness and accuracy for state space, invariants and operations. (CS UL = University Library case study (CS); FB = Flight Booking; FCUL, FCT/UNL, UL and UY are the different institutions.)

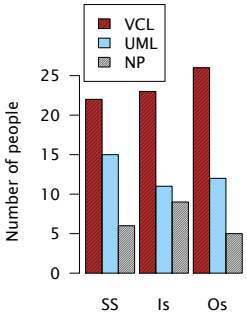
– Operations (Fig. 10f) highlight a non-significant VCL advantage ($M = .11$ $[.08, .14]$, $sd = .13$) to UML+OCL ($M = .09[.06, .11]$, $sd = .11$) — $MD = .02[0, .04]$, W $p = .07$ (ns), $d = .17[-.03, .4]$ (•).

8.2.3 Perceived Performance

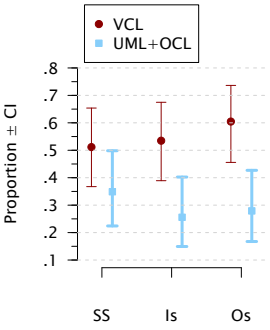
The debriefing questionnaire [8] asked about any perceived notation advantages in modelling. The results, corresponding to hypotheses $H_{7..9}$ of table 6, are given in Fig. 11, which contains a table of frequencies (Fig. 11a), a histogram (Fig. 11b) and a plot of proportions and their CIs (Fig. 11c). The results are as follows:

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

(a) Frequencies, significances and effects

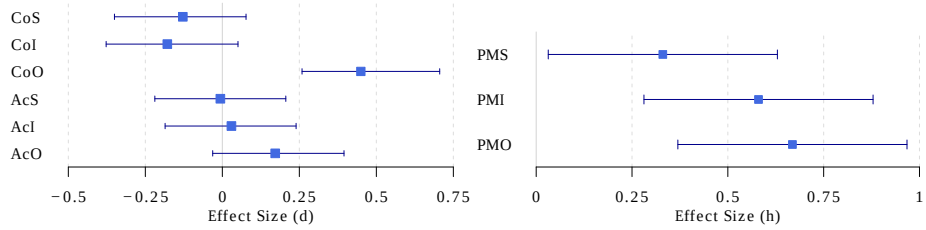


(b) Frequency distribution



(c) Proportions and CIs

Fig. 11: Notation perceived as providing best modelling performance (SS = State Space; Is = Invariants; Os = Operations; NP = No preference.)



(a) Objective performance (effect size d)

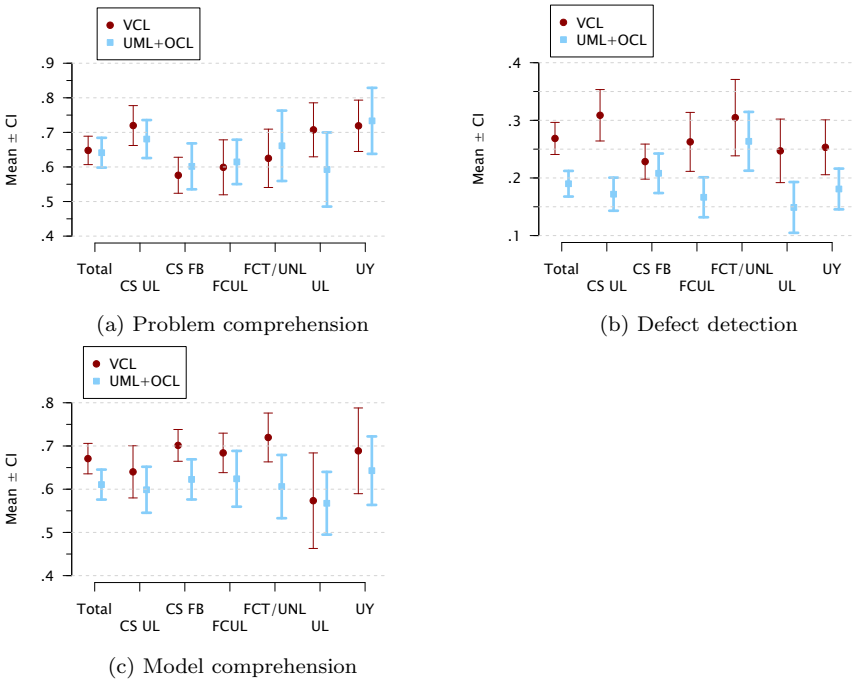
(b) Perceived performance (effect size h)

Fig. 12: Forest Plots of effect sizes and their CIs for modelling.

– In state-space (PMS), a higher, but non-significant proportion of subjects perceived a better performance with VCL (22 out 43 = .51[.39, .67]) than *UML+OCL* (15/43 = .35[.15, .4]) and *no preference* (6) — proportion difference (PD) = .16[−.11, .41], χ^2 p = .011 (*), BY q = .088 (ns), h = .33[.03, .63] (•+).
– In invariants (PMI), a higher, but non-significant proportion of subjects perceived a better performance with VCL (23/43 = .53[.39, .67]) than *UML+OCL* (11/43 = .26[.15, .4]) and *no preference* (9) — *PD* = .28[.02, .5], χ^2 p = .018 (*), BY q = .13 (ns), h = .58[.28, .88] (••).
– In operations (PMO), VCL (26/43 = .6[.46, .74]) significantly outperformed UML+OCL (12/43 = .28[.17, .43]) and *no preference* (5) — *PD* = .33[.05, .55], χ^2 p = .00034 (***), BY q = .0061 (**), h = .67[.37, .97] (••+).

8.2.4 Overall

Figure 12 presents forest plots of ESs for objective (Fig. 12a) and subjective (Fig. 12a) measures of modelling.



(a) Problem comprehension

(b) Defect detection

(c) Model comprehension

Fig. 13: Plots of means and CIs for model usage and comprehension (CS UL = University Library case study (CS); CS FB = Flight Booking CS; FCUL, FCT/UNL, UL and UY are the different experiment venues.)

8.3 Comprehension and Model Usage

The tasks of problem comprehension, defect detection and model comprehension resulted in objective and subjective performance measures.

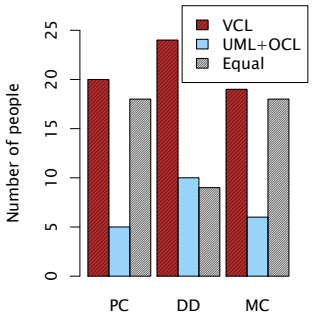
8.3.1 Objective Performance

The results, corresponding to hypotheses $H_{10..12}$ of table 6 and portrayed in Fig. 13, are as follows:

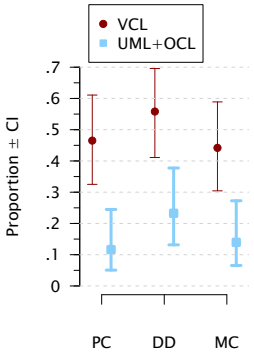
- In problem comprehension (PC, Fig. 13a), VCL ($M = .65[.61, .69]$, sd= .19) is quasi-equal to UML+OCL ($M = .64[.6, .68]$, sd= .2) — $MD = .01[-.04, .05]$).
- In defect detection (DD, Fig. 13b), VCL's ($M = .27[.24, .3]$, sd= .13) difference to UML ($M = .19[.17, .21]$, $sd = .1$) is highly significant — $MD = .08[.05, .1]$), W $p = 5 \times 10^{-7}$ (***), BY $q = 2 \times 10^{-5}$ (***), $d = .67[.41, .87]$ (●●+).
- In model comprehension (MC, Fig. 13c), VCL (M=.67[.64, .71], $sd = .16$) significantly outperforms UML+OCL ($M = .61[.58, .64]$, $sd = .16$) — $MD = .06[.02, .1]$), W $p = .0025$ (**), BY $q = .022$ (*), $d = .37[.09, .52]$ (●+).

	PC ns ●●●	DD ns ●●+	MC ns ●●+
VCL	20	24	19
UML	5	10	6
Equal	18	9	18
Total	43	43	43

(a) Frequencies, significances and effects

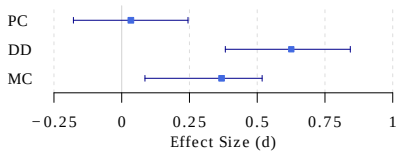


(b) Histogram

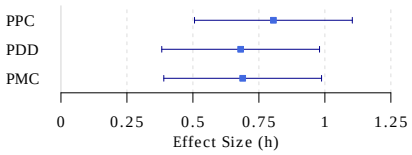


(c) Proportions and CIs.

Fig. 14: Notation perceived as providing the best performance in problem comprehension (PC), defect detection (DD) and model comprehension (MC).



(a) Actual performance (effect size d)



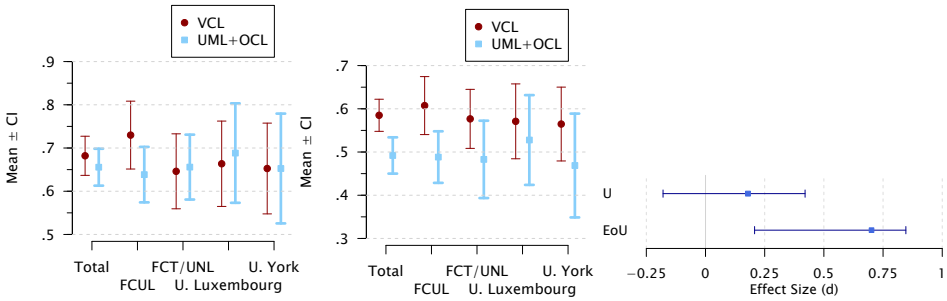
(b) Perceived performance (effect size h)

Fig. 15: Forest Plots of effect sizes of objective and perceived performance in problem and model comprehension (PC and MC) and defect detection (DD).

8.3.2 Perceived Performance

The results, corresponding to hypotheses $H_{13..15}$ of table 6, are described in Fig. 14 with a table of frequencies (Fig. 14a), a histogram (Fig. 14b), and a plot of proportions and CIs (Fig. 14c). The results are as follows:

- In PC, VCL ($P = .47[.33, .61]$) outperformed UML+OCL ($P = .12[.05, .24]$) non-significantly — $PD = .35[.13, .53]$, χ^2 $p = .0098$ (**), BY $q = .081$ (ns), $h = .81[.51, 1.1]$ (●●●).
- In DD, VCL ($P = .56[.41, .7]$) outperformed UML+OCL ($P = .23[.13, .38]$) non-significantly — $PD = .33[.06, .54]$, χ^2 $p = .0074$ (**), BY $q = .066$ (ns), $h = .68[.38, .98]$ (●●+).
- In MC, VCL ($P = .44[.3, .59]$) outperformed UML+OCL ($P = .14[.07, .27]$) non-significantly — $PD = .3[.08, .49]$), χ^2 $p = .026$ (*), BY $q = .17$ (ns), $h = .69[.39, .99]$ (●●+).



(a) Means and CIs (U) (b) Means and CIs (EoU) (c) Forest plot of effect sizes (d)

Fig. 16: Usefulness (U) and ease of use (EoU).

8.3.3 Overall

Figure 15 presents forest plots of objective and perceived performance in model usage and comprehension. Figure 15a portrays the Cohen's d ESs (Cohen's d) with CIs; Fig. 15b does the same with subjective measures and Cohen's h.

8.4 Usefulness and Ease of Use

Variables U and EoU aggregate perception scores (proportion) emerging from statements evaluated on Likert scale from 1 (strongly agree) to 5 (strongly disagree)[9]. The results, corresponding to hypotheses $H_{16,17}$ of table 6 and portrayed in the plots of Fig. 16, are as follows:

- In U (Fig. 16a), VCL ($M = .68[.64, .73]$, $sd = .15$) outperforms UML+OCL ($M = .66[.61, .7]$, $sd = .14$), but not significantly — $MD = .03[-.04, .09]$ (sd=.14), W $p = .19$ (ns), $d = .18[-.18, .42]$ (•).
- In EoU (Fig. 16b), VCL ($M = .58[.55, .62]$, sd=.12) significantly outperforms UML+OCL ($M = .49[.45, .54]$, sd=.14,) — $MD = .09[.04, .15]$ (sd=.14), W $p = .0026$ (**), BY $q = .022$ (*), $d = .7[.21, .85]$, (••+).

8.5 Usability

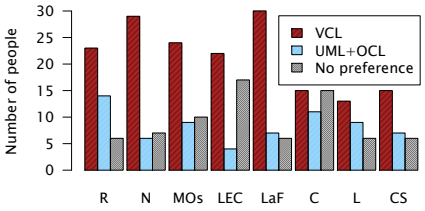
The usability results, corresponding to hypothesis $H_{18..25}$ of table 6, are portrayed in Fig 17, which contains a table of frequencies (Fig. 17a), a histogram (Fig. 17b), a plot of proportions and CIs (Fig. 17c), and a forest plot of ESs (Fig. 17d). They are as follows:

- In reading (R), VCL's proportion of .53 (CI [.39, .67]) against UML+OCL's .33 (CI [.19, .47]) yields a non-significant proportion difference (PD) of .21[−.07, .45] — χ^2 $p = .0064$ (**), BY $q = .066$ (ns), $h = .43[.13, .73]$ (••).

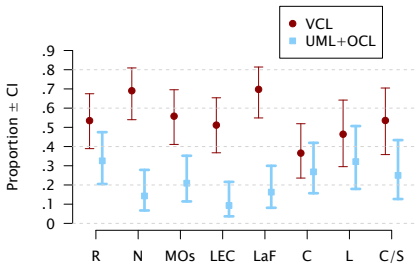
[9] For each response, the score is obtained by the formula $5 - x$ if the statement is positive and $x - 1$ if the statement is negative yielding a score on a scale 0..4 for each response.

	R ns ●●	N *** ●●●	MOs ns ●●+	LEC * ●●●	LaF *** ●●●	C ns ●+	L ns ●+	CS ns ●●
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

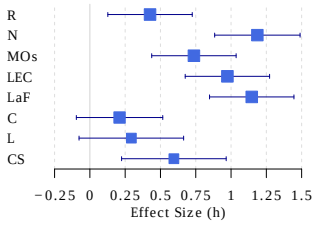
(a) Frequencies, significances and effects



(b) Frequency distributions



(c) Proportions and CIs



(d) Forest plot of effect sizes (h)

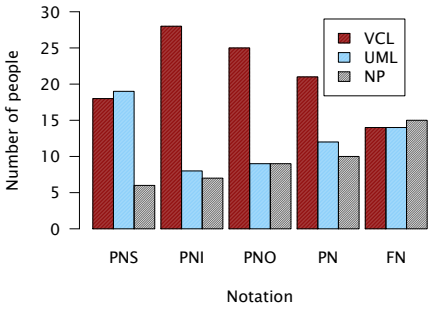
Fig. 17: Usability results derived from debriefing survey. (NP= No preference; R = Reading; N = Navigation; MOs = Maps and overviews; LEC = live error checking; LaF = Look and feel; C=Cohesion; L=Learnability; CS=Comfort Satisfaction.)

- In navigation (N), VCL's ($P = .69[.54, .81]$) difference to UML+OCL ($P = .14[.07, .28]$) is significant — $PD = .55[.29, .72]$), χ^2 $p = 6 \times 10^{-6}$ (***), BY $q = .00014$ (***), $h = 1.19[.88, 1.49]$ (●●●).
- In maps and overviews (MOs), VCL's ($P = .56[.41, .7]$) difference to UML+OCL ($P = .21[.11, .35]$) is not significant — $PD = .35[.09, .56]$), χ^2 $p = .0074$ (**), BY $q = .066$ (ns), $h = .74[.31, 1.16]$ (●●+).
- In live error checking (LEC), VCL ($P = .51[.37, .65]$) significantly outperformed UML ($P = .09[.04, .22]$) — $PD = .42[.2, .59]$, χ^2 $p = .0024$ (**), BY $q = .03$ (*), $h = .97[.68, 1.27]$ (●●●).
- In look and feel (LaF), VCL ($P = .7[.55, .81]$) is significantly higher than UML+OCL ($P = .16[.08, .3]$) — $PD = .53[.27, .72]$, χ^2 $p = 3 \times 10^{-6}$ (***), BY $q = 9 \times 10^{-}5$ (***), $h = 1.15[.85, 1.45]$ (●●●).
- In cohesion (C), VCL ($P = .37[.24, .52]$) non-significantly outperforms UML ($P = .27[.16, .42]$) — $PD = .1[-.14, .32]$, χ^2 $p = .68$ (ns), $h = .21[-.1, .52]$ (●+).
- In learnability (L), VCL ($P = .46[.3, .64]$) non-significantly outperforms UML+OCL ($P = .32[.18, .51]$) — $PD = .14[-.18, .43]$, χ^2 $p = .27$, $h = .29[-.08, .66]$ (●+).
- In comfort/satisfaction (CS), VCL's ($P = .54[.36, .7]$) advantage to UML ($P = .25[.13, .43]$) is not significant — $PD = .29[-.04, .55]$, χ^2 $p = .074$ (ns), $h = .6[.22, .97]$ (●●).

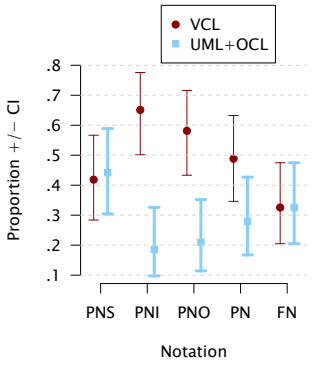
The usability analysis above together with Fig. 17d are consistent with the EoU results of the previous section. The spin-off study that compared the experiment's tools [9] highlighted that VCL underperformed UML+OCL in the *writing* criteria, albeit non-significantly; this suggests that, in certain circumstances, graphical editors are less convenient than their textual counter-parts [9].

	PNS ns ø	PNI ** ●●●	PNO * ●●+	PN ns ●●	NF ns 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

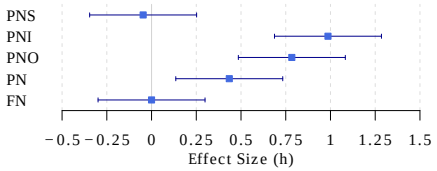
(a) Frequencies, significances and effects



(b) Frequency distributions



(c) Proportions and CIs



(d) Forest plot of effect sizes (h)

Fig. 18: Perceived preferred notation. (NP = no preference; PNS = preferred state space notation; PNI = preferred invariants notation; PNO = preferred notation for operations; PN = preferred notation overall; FN = notation to use in Future.)

8.6 Overall Perception

Participants' overall perception was appraised with respect to preferred notation, and positive and negative aspects.

8.6.1 Preferred Notation

The results, corresponding to hypotheses $H_{26..30}$ of table 6, are depicted in Fig. 18 with a table of frequencies with levels of significance and ES (Fig. 18a), a histogram (Fig. 18b), a plot of means and CIs (Fig. 18c) and a forest plot of ESs (Fig. 18d). They are as follows:

- In the state space (PNS), VCL ($P = .42[.28, .57]$) under-performs UML+OCL ($P = .44[.3, .59]$), but non-significantly — $PD = -.02[-.29, .24]$, χ^2 $p = .026$ (*), BY $q = .17$ (ns), $h = -.05[-.35, .25]$ (ø).
- In invariants (PNI), VCL ($P = .65[.5, .78]$) significantly outperforms UML+OCL ($P = .19[.1, .33]$) — $PD = .47[.2, .66]$, χ^2 $p = 6 \times 10^{-5}$ (***), BY $q = .0012$ (**), $h = .99[.69, 1.28]$ (●●●).

- In operations (PNO), VCL ($P = .58[.43, .72]$) significantly outperforms UML ($P = .21[.11, .35]$) — $PD = .37[.11, .58]$), χ^2 $p = .0026$ (**), BY $q = .03$ (*), $h = .78[.49, 1.08]$ (●●+).
- Overall, VCL's ($P = .49[.35, .63]$) advantage to UML ($P = .28[.17, .43]$) is not significant— $PD = .21[-.05, .44]$, χ^2 $p = .091$ (ns), $h = .43$, CI $[.14, .73]$ (●●).
- There is a tie in the notation to be used in the future (NF) — $P_{VCL} = P_{UML} = .33[.2, .47]$, $PD = 0[-.23, .23]$, $h = 0[-.3, .3]$ (ø).

These results endorse VCL's positive perceptions with its approach to invariants and operations being appraised favourably (Fig. 18d).

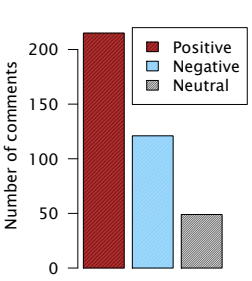
8.6.2 Positive and negative aspects

Individual comments to the several open questions of the debriefing survey were classified as positive, negative or neutral to VCL in comparison to UML+OCL. Figure 19 gives the results; it contains a table of frequencies (table 19a), a histogram (fig. 19b), a plot of proportions and CIs (Fig. 19c), and two histograms detailing positive and negative comments (Figs. 19d and 19e). The results of hypothesis H_{31} (table 6) are as follows:

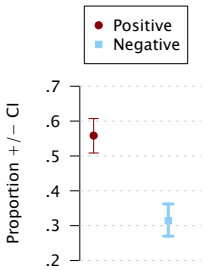
- From a total of 385 comments, 215 were positive ($P = .56[.51, .61]$), 121 negative ($P = .31[.27, .36]$) and 49 were neutral — table 19a and Fig. 19b.
- As signalled by Fig. 19c, positives significantly surpass the negatives — PD = $.24[.15, .33]$, χ^2 $p = 4 \times 10^{-24}$ (***), BY $q = 4 \times 10^{-22}$ (***),$h = .5[.4, .6]$ (●●).
- By dissecting the positive comments (Fig. 19d), we can see that understanding (U, 21), ease of use (oU, 20) and ease of finding errors (EFE, 19) were the most remarked. Participants also appraised positively VCL's modelling of behaviour (MB, 18) and invariants (MI, 14), and VCL's visualisations (V, 14), usability with respect to ease of access to information (EAI, 13) and navigability (N, 12), while appreciating VCL as an overall language (OL, 10). UML's Papyrus tool was perceived by many as difficult (TD, 10). Some participants appreciated VCL's overall modelling (M, 8), organisation (Or, 8), user interface (UI, 6), appealling (A, 5), and its sate modelling approach (MS, 5), while remarking UML+OCL's bad usability (BU, 6). A few participants found it comfortable to work with VCL (Ct, 4), appreciated VCL's cohesion (Cn, 4) and capacity to provide overviews (Ov, 4) while remarking that OCL is difficult (D, 4).
- In terms of VCL's negatives (Fig. 19e), the most remarked aspects were UML's familiarisation (F, 16), the fact that UML is more know (MK, 9) and VCL's bad usability (BU, 9). Some participants remarked UML's ease of use (EU, 8), modelling of state (MS, 8) and behaviour (MB, 7), and cohesion (Cn, 7), while remarking VCL's cumbersome modelling of behaviour (CB, 8) and cumbersome editing (Ed, 7). Some participants appraised positively UML's Papyrus tool (T, 6) and UML's understanding (U, 5), while recognising that VCL's tool is difficult (TD, 6). A few praised UML as an overall language (OL, 4), its capacity to provide overviews (Ov, 3) and its expressivity (Ex, 3), and that UML is okay and does the job (Ok, 3), while emphasising that they felt comfortable using UML (Ct, 3) and that it is easy to find errors in UML models (EFE, 3).

Positive	215
Negative	121
Neutral	49
Total	385

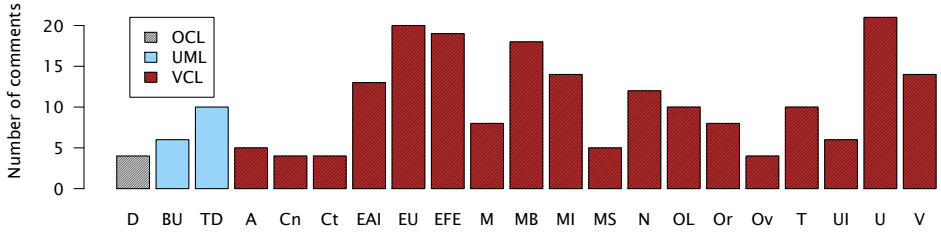
(a) Frequencies



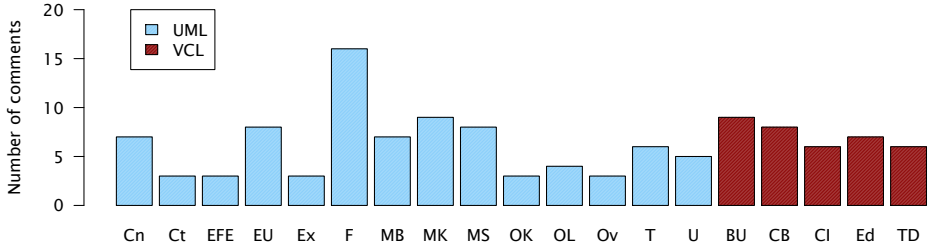
(b) Histogram



(c) Proportion and CIs



(d) Positive comments

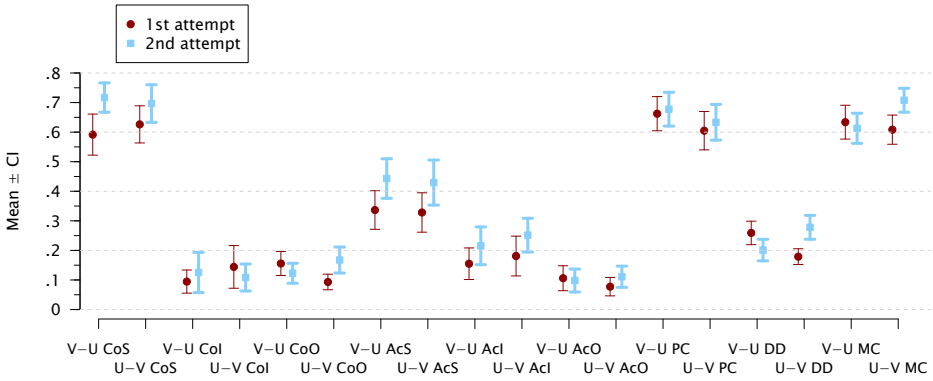


(e) Negative comments

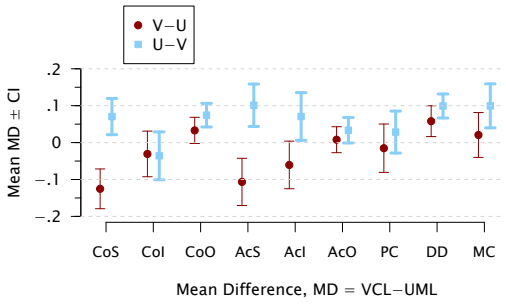
Fig. 19: Perceived positives and negatives of VCL in comparison to UML+OCL. (A = appealing; BU = bad usability; CB = cumbersome modelling of behaviour; CI = cumbersome modelling of invariants; Cn = cohesion; Ct = comfort; D = difficult; EAI = easy to access information; Ed = editing; EU = ease of use; EFE = easy to find errors; Ex = expressivity; F = familiarity; M = modelling; MB = modelling behaviour; MI = modelling of invariants; MK = more known; MS = modelling of state; N = navigability; Ok = it's okay; OL = overall language; Or = organisation; Ov = overviews; T = tool; TD = tool difficult; UI = user interface; U = understanding; V = visualisation.)

8.7 Learning effects

Fig. 20 depicts an analysis of learning effects typical of crossover designs [55]. Figure 20a contrasts first and second attempts at the different experiment tasks for the orders V-U (VCL followed by UML) and U-V (UML followed by VCL). For four (out of nine) measures, namely, completeness of state space (CoS), accuracy of state space (AcS), accuracy of invariants (AcI) and program comprehension (PC), there is a performance improvement on the second attempt (a learning effect)



(a) The two attempts at the experiment tasks in relation to the orderings V-U and U-V



(b) Mean differences between orderings V-U and U-V for each task

	p	d
CoS	$3 \times 10^{-7}_{***}$	$1.18[.58, 1.31]_{\bullet\bullet\bullet}$
CoI	$.84_{ns}$	$-.02[-.32, .28]_{\emptyset}$
CoO	$.12_{ns}$	$.38[-.04, .57]_{\bullet+}$
AcS	$2.8 \times 10^{-5}_{***}$	$1.06[.46, 1.15]_{\bullet\bullet\bullet}$
AcI	$.0032_{**}$	$.63[.15, .78]_{\bullet\bullet+}$
AcO	$.53_{ns}$	$.23[-.15, .45]_{\bullet}$
PC	$.084_{ns}$	$.21[-.1, .5]_{\bullet}$
DD	$.052_{ns}$	$.34[-.02, .59]_{\bullet+}$
MC	$.1_{ns}$	$.4[-.02, .59]_{\bullet\bullet}$

(c) Analysis of paired differences of orders V-U and U-V

Fig. 20: Learning effects (Legend: V-U = VCL followed by UML; U-V = UML followed by VCL).

independent of the used notation. For completeness of invariants (CoI), there are improvements only when UML is used in the second attempt. For four measures, completeness of operations (CoO), accuracy of operations (AcO), defect detection (DD) and model comprehension (MC), there are improvements only when VCL is used in the second attempt.

Figure 20b pictures the means and CIs of the paired differences of both V-U and U-V for each task measure — $PD = V_{VCL} - V_{UML}$. Table 20c gives the calculated p-values and ESs of PD(V-U) and PD(U-V). We observe a significant effect for completeness of state space (CoS), accuracy of state space (AcS) and accuracy of invariants (AcI). The remaining differences are not significant; ESs tend to be small, being medium only for model comprehension (MC). Overall, there is a general tendency for the paired difference to be higher when VCL is used on the second attempt. This suggests that due to VCL's novelty, participants' VCL proficiency grows as subjects learn by doing the experiment's tasks.

9 Threats to Validity

This section discusses threats to the validity of the experiment reported here, including conclusion, construct, internal and external validity.

9.1 Conclusion Validity

Conclusion validity is concerned with the relation between treatment and outcome, and the statistical results. Following from the debate on null-hypothesis significance testing (NHST) [34,37,83] and recommendations of high-ranked social science journals, the analysis supplemented NHST with levels of uncertainty, plausibility and magnitude; measures of central tendency (means and proportions) were accompanied by confidence intervals (CIS) and effect sizes (ESs). As highlighted in section 6.2, null hypotheses are rejected based on adjusted p-values only, calculated using the false discovery rate [22] and the method of Benjamini and Yekutieli [23].

The statistical analysis strived for robustness. We used non-parametric tests and avoided breaching tests' assumptions. Continuous variables were analysed using the non-parametric Wilcoxon test as it is not dependent on normality assumptions; Wilcoxon calculated p-values are consistent with parametric t-tests and the robust trimmed means test [115] suggested by Wilcox and Keselman [111]. The χ^2 test, used to assess categorical variables, is robust with respect to the distribution of the data, but all categorical variables were found to be normal by Pearson's test. All statistical testing results are given in [8]; here we provide Wilcoxon, χ^2 and BY p-values only.

For ESs, we complemented the classical Cohen d with the more robust alternatives of Algina et al [2] and Wilcox and Tian [112]. All calculated ESs, given in full in [8], were found to be consistent; here we focussed on Cohen's d and h.

9.2 Construct Validity

Construct validity concerns how measurements are affected by factors related to experimental settings. It relates to training, case studies, measurement instruments, including the defects seeded, and the different questionnaires. Table 7 presents statements drawn from the debriefing survey, accompanied by levels of significance and effect, that supports the analysis that follows. Likewise for Figure 21, which depicts performance hindering factors.

The training concentrated on the challenging modelling tasks and relied on participants' prior exposure to software modelling (section 4.5). It tried to: refresh or enhance design skills and familiarise participants with the modelling tools within the time available. Many participants felt that (table 7, *training*): the training was insufficient (T1, table 7), more training would have improved performance (T5), training was a major performance-hindering factor (Fig. 21, LTr), they lacked knowledge and experience (Fig. 21, LKE), they felt prepared with UML (T2), but not with VCL or OCL (T3 and T4, respectively). This suggests the following: (a) the experiment's tasks were challenging because many participants were unhappy with their performance; (b) participants' prior exposure to UML (confirmed by

Id	Question	N	Mean	SD
Training				
T1	The training allowed me to carry out the tasks of the experiment competently	43	$3.21[2.91, 3.51]_{***,\bullet+}$	.99
T2	I had enough training in UML	43	$2.56[2.24, 2.88]_{*,\bullet\bullet}$	1.08
T3	I had enough training in OCL	43	$3.67[3.38, 3.96]_{***,\bullet\bullet\bullet}$	.97
T4	I had enough training in VCL	43	$3.35[3.09, 3.61]_{*,\bullet\bullet}$	.87
T5	I would have performed better if I were given more hours of training	43	$1.63[1.39, 1.86]_{***,\bullet\bullet\bullet}$	.79
Case Studies				
CS1	I had enough time to read through the requirements descriptions of the given case studies	43	$2.02\ [1.74, 2.31]_{***,\bullet\bullet\bullet}$	.96
CS2	I fully understood the systems described in the requirements documents	43	$2.21[1.97, 2.45]_{***,\bullet\bullet\bullet}$	.8
CS3	In general, I had no problems carrying out the experiment's tasks	43	$2.79[2.52, 3.06]_{NS,\bullet+}$	.91
CS4	The instructions were clear and easy to follow	43	$1.77[1.56, 1.97]_{***,\bullet\bullet\bullet}$	.68
CS5	The university library system was fairly easy to understand	43	$2.02[1.79, 2.25]_{***,\bullet\bullet\bullet}$	.77
CS6	The flight booking system was fairly easy to understand	43	$2.14[1.87, 2.41]_{***,\bullet\bullet\bullet}$	.89
CS7	The university library case study was realistic; a good example of a real-world case study	7	$1.57[.99, 2.15]_{*,\bullet\bullet\bullet}$	.79
CS8	The flight booking case study was realistic; a good example of a real-world case study	7	$1.86[1.19, 2.52]_{NS,\bullet\bullet\bullet}$	.9
Defect Detection				
DD1	I fully understood the task	43	$1.81[1.61, 2.02]_{***,\bullet\bullet\bullet}$	.7
DD2	I was not sure what type of errors I was looking for	43	$3.02[2.65, 3.4]_{NS,\emptyset}$	.03
Modelling Tools				
MT1	The modelling tools were fairly stable	43	$2.23[1.96, 2.51]_{***,\bullet\bullet\bullet}$	.92

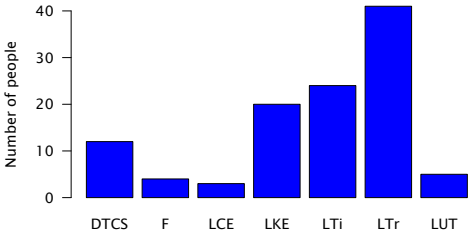
Table 7: Debriefing survey statements related to training, case studies, defect detection and modelling tools, rated on a Likert scale from 1 (strongly agree) to 5 (strongly disagree). Columns: mean of response with 95% CIs, standard deviation (SD). Statement are compared against neutrality to derive significance and effect.

the results of T2 in comparison to T3 and T4) could have biased the results in favour of UML, but the results are positive to VCL, hence: (i) although seen as insufficient, the training seems to have worked, and (ii) VCL accommodates UML-trained practitioners; (c) training was insufficient for the modelling of invariants and operations as most participants lacked prior training.

The experiment's case studies (CSs), exemplars of real-world software systems similar to CSs used in the teaching of software design, do not favour one notation over the other, and are from familiar application domains (a university library and a flight booking system). They laid the foundation for tasks that are both feasible and challenging, but not overwhelmingly complex. Participants recognised many of these characteristics in the CSs, as evidenced by table 7 (case studies); most subjects felt that: (a) they had enough time to read the use case narratives (CS1), which (b) they understood (CS2), (c) the case studies were fairly easy to understand (CS4 and CS5). Many subjects felt that: the tasks were carried with difficulties (CS3 in table 7 and DTCS in Fig. 21). A few subjects lacked a

DTCS	12
F	4
LCE	3
LKE	21
LTi	25
LTr	41
LUT	5

(a) Frequencies



(b) Histogram

Fig. 21: Perceived performance hindering factors derived from the debriefing survey. (DTCS = Difficult tasks and case studies; F = Fatigue; LCE = Lack of concentration and engagement; LKE = Lack of knowledge and experience; LTi = lack of time; LTr = lack of training; LUT = Lack of understanding of tasks.)

clear understanding of the tasks (Fig. 21, LUT). York participants ($N = 7$) found the CSs to be realistic (CS7, CS8). Overall, this suggests that: (i) the tasks were challenging, (ii) many participants lacked prior knowledge for such a challenging experiment, (iii) although the training worked somehow, it was not able to fill the knowledge gaps of most subjects.

Lack of time (Fig. 21, LTi, 24 out of 43) was a performance obstacle. Limited time and CS's reasonable complexity were countered with starting models to be completed within the allotted times. Despite this, the tasks on modelling of operations and invariants, and problem comprehension, were not entirely satisfactory. Modelling of invariants had to be done together with the state space within 30 minutes, which proved short; modelling of operations was too difficult for a 35-minute task. Task's short duration, lack of training and inherent difficulty, explain the low results obtained in these tasks.

One of the authors marked the models constructed by the participants. To minimise bias and maximise objectivity, a systematic, repeatable and divide-and-conquer scoring approach, detailed in [8], was pursued. Both completeness and accuracy relied on a requirements-based marking breakdown made up of individual items that are fairly objective and with little room for subjectivity. The completeness scoring scheme focused on the modelling required by each requirement; the accuracy scoring scheme consisted of test-cases, expressed as snapshots or object diagrams based on an approach outlined in [19,4]. Marking both completeness and accuracy involved going through each item of the marking scheme.

The seeded defects, chosen to avoid both favouring any of the notations and being obvious, were scattered evenly across state space, invariants and operations (see section 5.1). Defect detection (DD) was designed to be intuitive and challenging, which was acknowledged by the participants (DD results in table 7); an overwhelming majority of them understood the task (DD1, in table 7) and many found it challenging because the errors were not obvious (DD2), which is endorsed by DD's modest final scores: .27 for VCL and .19 for UML+OCL.

The model comprehension questions were selected to be of a certain level of complexity and to ensure a reasonable level of coverage. They were articulated to avoid bias in favour of one treatment over the other; the multiple-choice questions eased marking. The debriefing survey was designed to capture the perceptions of

subjects with respect to the experiment experience. It contained questions that targeted experiment hypothesis related to subjective assessments. To avoid bias, and to increase the reliability of the collected data, we followed guidelines of questionnaire design [85].

The modelling tools could bias or confound the results. We used tools built atop the same platform, Eclipse, to ensure a certain degree of similarity. Papyrus is a major UML Eclipse-based tool with a significantly larger user-base than VCL's tool, VCB. Despite some criticisms, both tools were seen as being stable (table 7, *modelling tools*). A spin-off survey, tied to the experiment presented here [9], compared these two tools with a focus on usability; the results, favourable to VCB, concluded that participants could not always discern between tool and language. Nevertheless, VCB is the implementation of VCL's language and its underlying ideas. The fact that VCB was competitive with Papyrus, a major UML tool, testifies to the quality of both VCL and VCB. Certain VCB features may be superior to Papyrus, but that per-se does not account for VCL's positive results presented here and in [9]. Both tools, built on top of Eclipse's modelling infrastructure, have considerable room for improvement.

9.3 Internal Validity

Internal validity threats are related to external factors that affect the outcomes of the experiment, but are not a consequence of the studied treatment. Table 8 presents statements drawn from the debriefing survey, accompanied by levels of significance and effect, that aids the analysis presented here.

The experiment's volunteer basis is a possible internal threat. Usually, experiments resort to blocking to counter against variability in the capability of subjects. In our setting, this was, by and large, infeasible; assignment to a group was mainly decided on the basis of a subject's availability. Section 7 highlights a slight and non-significant imbalance in the proficiency scores of both experimental groups.

Information exchange did not affect the experiment: (i) all participants were working in parallel on the same case study, (ii) group sessions would run separately, but on the same day (morning or afternoon) and (iii) participants were busy. There was hardly any opportunity or motivation for exchanging information.

The experiment's crossover design suffers from known *carry-over* effects [55]. Here, the most relevant carry-over effect is the learning accrued from carrying out tasks on the same system using the two languages; modelling using one language may give rise to a learning effect leading to an improved performance when modelling with the next language. To balance the experiment and counter against such learning effects, the order of the two notations with respect to the two case studies was permuted for each group — a group starting with VCL on university library (UL) would start with UML+OCL on flight booking (FB), and vice-versa. We followed a 2^2 factorial design (case study is a factor) with the case study tasks being undertaken in parallel to: (a) avoid case study difficulty as a confounding effect [10], (b) counter against exchanges of information, and (c) have more experiment

[10] The different levels of difficulty was acknowledged by the participants in the debriefing survey; 27 out of 43 participants said that FB was more complex ($p = .63[.48, .76]$), 11 found that UL was more complex ($p = .26[.15, .4]$) and 5 found the two case studies to be similar

Id	Question	N	Mean	SD
E1	I liked the experiment	35	$1.89[1.61, 2.16]_{***,\bullet\bullet\bullet}$	.83
E2	The experiment was interesting	43	$1.93[1.66, 2.2]_{***,\bullet\bullet\bullet}$	.91
E3	During the experiment I was motivated, committed and I felt challenged	43	$2.23[1.94, 2.52]_{***,\bullet\bullet\bullet}$	.97
E4	I felt it was a good learning experience	43	$1.95[1.7, 2.21]_{***,\bullet\bullet\bullet}$	.84

Table 8: Debriefing survey statements related to the overall experiment rated on a Likert scale from 1 (strongly agree) to 5 (strongly disagree). Columns: mean of response with 95% CIs, standard deviation (SD). Each statement is evaluated against neutrality to derive significance and effect size.

practice (a total of 4 rounds) because software design is challenging, the training was somehow insufficient (see discussion on training in section 9.2 above) and any practice improvements would only lead to more interesting results and better founded opinions for the debriefing survey. To counter against further carry-over effects, participants were not given feedback on their performance. The analysis of order of notation (section 8.7) detected a few significant learning effects with large or medium effect sizes, however, this does not affect the paper's main results: modelling of operations, defect detection and model comprehension.

Lack of engagement was refuted by the debriefing survey (table 7): participants liked the experiment (E1), finding it interesting (E2), somehow challenging (E3) and positive from a learning perspective (E4). One of the options to the question "please indicate the reasons for your lack of motivation if you felt somehow demotivated" was "I do not really see the need for software design modelling in software engineering", which was selected by only one participant (out of 43).

9.4 External Validity

External validity, concerned with the generalisability of the results, is a recurrent issue in software engineering research [51]. Given that we want to extrapolate our results to software engineering practice, a question that emerges refers to the experiment's degree of realism [102]. In software engineering controlled experiments there is often an issue of scalability. Due to time constraints, size of case studies and tasks are reduced, and this may break the connection to industrial realities which usually deal with larger problems. We alleviated this issue with case studies that are neither too small nor trivial, and are realistic. This was acknowledged by many participants (see section 9.2, above). To ensure the feasibility of the modelling tasks within allotted times, participants were given partial models that they were required to complete.

Another problem is the degree of representativeness with respect to software engineering practitioners [43]. The participant cohort was culturally diverse; we ran the experiment in universities of three European countries, involving participants from across the globe. Participants were students who may be perceived as

of complexity ($p = .12[.05, .24]$) with the difference between FB and UL being significant — $PD = .37[.09, .59]$, χ^2 p-value = .00012 (***), ES $h = .77[.47, 1.07]$ (●●+).

unrepresentative of industrial professionals; however, many postgraduate participants had previous industrial experiences, and many were about to embark on new professional careers in industry. The experiment results attest to the cohort's degree of representativeness; a few individuals performed remarkably high, others noticeably low. Collectively, the average results on state space modelling and other model usage and comprehension tasks show that participants were able to undertake these required tasks, even though they were challenging. The low results on the modelling of invariants and operations are more due to the experiment's time constraints, the inherent difficulty of these tasks and the limited time available for training, rather than the cohort's overall ability. We see our cohort as a reasonably competent workforce, which is reflective of industrial settings applying design modelling. Empirical studies failed to find significant differences in performance between students and professionals [63,21,100]; a recent study found out that professionals appear to perform better in tasks they are accustomed to, but there is no difference when it comes to using new approaches or technologies [100].

A criticism of software engineering controlled experiments concerns pen-and-paper tasks which are not seen as reflecting modern-day realities. The experiment presented here used Eclipse tools, a popular platform for modern-day software development that we see as reflective of current practice.

We see the results presented here as generalisable to general graphical modelling, even though only two languages are examined. This is because VCL is a suitable representative of a largely visual software design language (section 2.4) and UML+OCL is a suitable baseline.

10 Related Work

This is the first empirical study that investigates the benefits of a modelling language capable of expressing predicates graphically, a pre-requisite to the diagrammatic expression of: (i) system invariants, (ii) and operations as contracts made up of pre- and post-conditions. Other languages with this capability, Visual OCL [27, 41] and Augmented Constraint Diagrams [48], lack empirical scrutiny.

Table 9 summarises related empirical studies. This paper covers several aspects of previous work:

- It covers comprehension, a focus of many studies [89,90,108,104,92,93] to investigate whether either notation or modelling per se are fulfilling their aims. This paper insists on end-user comprehension (either through model comprehension or defect detection tasks), but going beyond to explore the problem comprehension gained from modelling.
- It goes beyond data modelling, sharing with [68] the emphasis on the modeller perspective, with Otero and Dolado [**?**] the focus on dynamic modelling and with Briand et al [29,28] the focus on constraints and design by contract, but exploring novel graphical notations to the modelling of invariants and operations not covered by any other study.

Certain interesting aspects of related work are unexplored here:

- The impact of UML design on maintenance involving either code or design changes [**?**,20], which also delves into model comprehension as understanding is often a precondition to accomplishing the changes that are required.

Study	Goal	Perspective	Tasks	Participants
Kim and March [68]	Investigate effectiveness of data (or state space) modelling notations on comprehension and modelling.	end-user, modeller	(i) modelling from textual requirements, (ii) questionnaire on model comprehension, (iii) detecting model defects against textual requirements.	28 postgraduate students (users) and 26 professionals (modellers).
Moody [78]	Investigate comprehension of large data modelling mechanisms	end-user	(i) questionnaire on model comprehension, (ii) defect detection	60 students
Purchase et al [89, 90]	Investigate comprehension of variants of UML class and collaboration diagrams.	end-user	(i) identifying correct and incorrect diagrams against textual requirements	34 students and 5 researchers in [90], 35 students in [89]
Otero and Dolado [**?**]	Investigate comprehension of UML dynamic notations (statecharts, and sequence and communication diagrams)	end-user	(i) model comprehension questionnaires	18 final year undergraduates
Tilley and Huang [**?**]	Investigate effectiveness of UML in facilitating program understanding.	end-user, programmer	(i) a questionnaire on impact of changes to a system from code and UML-based documentation.	15 expert academics
Torchiano [108]	Study comprehension of UML class models accompanied by object diagrams.	end-user	(i) model comprehension questionnaires, (ii) debriefing survey	17 masters students
Briand et al [29]	Study comprehension of UML-based with OCL	end-user	(i) questionnaires on model comprehension and (ii) maintenance (impact of changes to a model), (iii) defect detection and (iv) debriefing survey.	38 final-year undergraduates
Arisholm et al [20]	Investigate effectiveness of UML models with respect to maintenance.	end-user, programmer	(i) code and (ii) design modifications on given design and code, (iii) post-task survey	96 undergraduate students.
Staron et al [104]	Investigate comprehension of stereotyped UML class models	end-user	(i) model comprehension questionnaires, (ii) debriefing survey	44 students and 4 industry professionals
Ricca et al [92, 93]	Investigate comprehension of stereotyped UML class models	end-user	(i) model comprehension questionnaires, (ii) debriefing survey	51 subjects among under- and post-graduates, and researchers.
Briand et al [28]	Study system sequence diagrams and natural language contracts on quality of UML-based models.	end-user, modeller	(i) modelling	223 final year undergraduates

Table 9: Empirical studies related to the work presented here.

– Coarse-grained modelling [78, 44, 3], which is related to VCL's coarse grained modularity approach inspired by aspect-oriented modelling [15].

No other work in the literature has looked into the graphical expression of invariants and operations. This paper explores this problematic through a comparison of VCL against OCL, which expresses invariants and operations textually. It examines VCL's novel graphical approach with respect to end-user and modeller understanding, modelling effectiveness and usability.

11 Conclusions

Visual modelling has always been part of software engineering [31,96,97]. Graphical design approaches have been advocated for nearly three decades [58,59]. Nevertheless, our knowledge of visual modelling is sketchy. We know that issues of syntax are vital to the usability of diagrammatic notations, seen as de-facto languages of software engineering practice [79]. However, the alleged benefits of visual modelling constitute a patchy region made up of many gray areas lacking empirical scrutiny. Is visual modelling a good idea?

This article shed some light on this general question. If data modelling is well studied and reasonably convincing, the same is not true for other more intricate aspects of software design. This paper pursues an answer to the following question: can we model effectively more complicated aspects of a software design, such as constraints and operations, graphically? If the research on the visual expression of predicates and system dynamics has shown that a largely visual approach is possible, we are still largely unsure on whether graphics are any better than text.

This papers delves into this question through a controlled experiment carried out four times, which by studying VCL's effectiveness as a visual language tries to draw more general conclusions about graphical modelling. VCL is largely graphical; different modelling aspects are expressed largely diagrammatically, including invariants and operations. It is a suitable representative of largely diagrammatic languages. The experiment compares VCL against the standard UML and its OCL satellite notation, which, together, champion an approach to design modelling that is partially graphical with invariants and operations expressed textually in OCL. The experiment involved 43 students from four universities in three countries who received training in UML, OCL and VCL. The comparison focussed on: (i) modelling of state space, invariants and operations (RQ1); (ii) problem comprehension (RQ2); (iii) model defect detection (RQ3); (iv) end-user model comprehension (RQ3); (v) usefulness and ease of use (RQ4); (vi) usability (RQ5); and (vii) overall appraisal (RQ6). Aspects (i) and (ii) take the perspective of the modeller or designer, (iii) and (iv) of a design end-user, and (v)–(vii) of a general software engineer or computer scientist. Careful attention was paid to ensure that the observed trends were due to the notations and not other extraneous factors.

A relevant result of this paper is that VCL modelling of operations got better results than OCL. Individuals performed significantly better using VCL with respect to completeness, but not accuracy. A significant proportion of subjects perceived a better VCL performance, VCL was the preferred notation for modelling operations with a significant difference and behavioural modelling was perceived as a major positive aspect of VCL (Fig. 19d). Therefore, results suggest benefits of diagrammatic modelling of operations in a VCL style.

Results are unclear for the remaining modelling aspects. In state-space modelling, VCL's graphical improvements were appreciated by an interesting minority, but most remained agnostic to them, possibly due to the familiarity of widespread UML class diagrams (Fig. 19e). In the objective measures, VCL failed to provide an improvement. Nevertheless, VCL's conservative approach with respect to UML modelling paid off: with some training participants transposed their prior UML knowledge into VCL. Hence, on the one hand, the familiarity of UML class diagrams is hard to beat, but, on the other hand, participants transposed knowledge across languages becoming accustomed to the differences in syntax.

RQ	Findings
RQ1	On modelling of state space, invariants and operations: • VCL's structural diagrams (SDs) were quasi-equal to UML class diagrams (variables CoS, AcS and PMS). • On invariants, no significant VCL benefits were encountered. • On operations, significant VCL benefits were encountered in completeness (CoO), but not accuracy (AcO). A significant proportion of subjects perceived a better VCL performance (PMO).
RQ2	On the modeller's comprehension of modelled problem, no significant differences were encountered in both objective (PC) and subjective measurements (PPC).
RQ3	On model usage, significant VCL benefits were encountered in objective measurements of defect detection (DD) and model comprehension (MC).
RQ4	In usefulness (U), no significant VCL benefits were encountered. In ease of use (EoU), VCL was perceived as being significantly better than UML+OCL.
RQ5	Usability was analysed from different angles. Significant VCL benefits were found for navigation (UsN), live error checking (LEC) and look and feel (LF).
RQ6	On the overall perception: • A largely significant proportion of participants preferred VCL as a notation to express invariants (PNI) and operations (PNO). • In the appraisal of positive and negative aspects (Appr), VCL was appraised favourably as positives significantly surpassed negatives.

Table 10: Summary of the experiment's findings per research question (RQ).

In the modelling of invariants, VCL failed to provide significant improvements. However, a non-significant proportion of participants perceived a VCL improvement, a significant proportion of participants chose VCL as the preferred notation for invariants, and VCL modelling of invariants was perceived as a major positive aspect (Fig. 19d). These positive results, together with the low scores in the modelling of invariants and operations, suggests the need for a better experimental apparatus, with improved training and more time devoted to carry out these more complex tasks — as remarked by the participants' appraisal of the training (section 9.2).

In model usage, the results signal a VCL improvement. In defect detection (DD), VCL's objective performance was significantly better, which is consistent with the way participants perceived their performance; ease of finding errors in VCL models was a frequently occurring positive aspect of VCL (see Fig. 19d). In model comprehension (MC), VCL was significantly better, which is consistent with the way participants perceived their performance; comments concerning understanding, ease of use, easy to access information were among the most frequently occurring VCL positive aspects (see Fig. 19d).

In three usability criteria, navigation, live error checking and look and feel, VCL outperformed UML+OCL significantly. It is interesting to relate these criteria to the theories of notation design considered here: physics of notations (PoN) [79] and cognitive dimensions of notations (CDN) [**?**,54, 24]. Navigation is related to PoN's principle of cognitive integration (also maps and overviews, rated well but without significance); live error checking is related to the CDN's error proneness and hard mental operations, as VCL's tool warns users when they write something meaningless which aids reasoning and avoids mistakes; and look and feel is related

to PoN's principles of semiotic clarity, perceptual discriminability and complexity management — a result of VCL's syntactic clarity, visual expressivity and overall tidiness. This suggests that, with respect to these theories of visual notation design, VCL and its tool appear to be better than UML+OCL and Papyrus.

Both notations were perceived as equally useful, but VCL was largely perceived as more easy to use. VCL was also the preferred notation for invariants and operations by a significant proportion. Finally, VCL was also highly appraised as positive in comparison to UML. This appears to endorse VCL's better model usage results and VCL's overall graphical approach.

The findings presented here are summarised in table 10 for each research question (section 3.2). Overall, results suggest usability benefits of graphical software design, which was clear in model usage and more modest in modelling. Participants responded well to VCL's novel graphical notations, providing empirical evidence to motivate further research on diagrammatic modelling.

The results presented here should not be regarded as a claim of absolute scientific truth, but rather as a contribution to a research question. The stronger results need to be confirmed through replication, and the weaker results together with the new questions spurred by the paper's analysis require further experimentation.

Acknowledgements A large part of the research presented in this paper was done during Amálio's post-doctoral research work funded by the University of Luxembourg. Lionel Briand's research was undertaken, in part, thanks to funding from the Canada Research Chairs program. We thank the reviewers for their suggestions and the way they challenged the paper; this led to many improvements that resulted in a better article. We thank Christian Glodt for his help with the VCL tool, and all participants of the controlled experiment presented here at the Faculty of Sciences of the University of Lisbon (FCUL), Faculty of Sciences and Technology of the New University of Lisbon (FCT/UNL), University of Luxembourg and University of York (UY). Several people at the host institutions made the controlled experiment possible, namely, Isabel Nunes (FCUL), João Araújo (FCT/UNL), Fiona Polack (UY), Antónia Lopes (FCUL) and Ana Moreira (FCT/UNL).

References

1. Abrial, J.R.: The B book: assigning meaning to programs. Cambridge University Press (1996)
2. Algina, J., Keselman, H.J., Penfield, R.D.: An alternative to cohen's standardized mean difference effect size: a robust parameter and confidence interval in the two independent groups case. Psychological Methods **10**(3), 317–328 (2005)
3. Ali, S., Yue, T., Briand, L.: Does aspect-oriented modeling help improve the readability of uml state machines? Sofwtare and Systems Modeling **13**(3), 1189–1221 (2014)
4. Amálio, N.: Generative frameworks for rigorous model-driven development. Ph.D. thesis, Dept. Computer Science, Univ. of York (2007)
5. Amálio, N.: VCL model of the secure simple bank case study. Tech. Rep. TR-LASSY-11-05, Univ. of Luxembourg (2011). `http://bit.ly/q1LrPj`
6. Amálio, N.: The VCL model of the Barbados crisis management system. Tech. Rep. TR-LASSY-12-09, Univ. of Luxembourg (2012). `http://bit.ly/W5C8ZY`
7. Amálio, N.: Relaxing behavioural inheritance. In: Refine 2013, *EPTCS*, vol. 115, pp. 68–83 (2013)
8. Amálio, N., Briand, L., Kelsen, P.: An empirical evaluation of visualisation in software design modelling: the VCL vs UML+OCL experiment. Tech. Rep. TR-LASSY-13-05, LASSY, Univ. of Luxembourg (2013). URL `http://bit.ly/2uVXGkz`
9. Amálio, N., Glodt, C.: A tool for visual and formal modelling of software designs. Science of Computer Programming **98, Part 1**, 52 – 79 (2015). DOI 10.1016/j.scico.2014.05.002

10. Amálio, N., Glodt, C., Kelsen, P.: Building VCL models and automatically generating Z specifications from them. In: FM 2011. Springer (2011)
11. Amálio, N., Glodt, C., Pinto, F., Kelsen, P.: Platform-variant applications from platform-independent models via templates. ENTCS **279**(3), 3–25 (2011)
12. Amálio, N., Kelsen, P.: Modular design by contract visually and formally using VCL. In: VL/HCC 2010 (2010)
13. Amálio, N., Kelsen, P.: VCL, a visual language for abstract specification of software systems formally and modularly (short paper). In: Diagrams 2010, *LNAI*, vol. 6170. Springer (2010)
14. Amálio, N., Kelsen, P., Ma, Q.: Specifying structural properties and their constraints formally, visually and modularly using VCL. In: EMMSAD 2010, *LNBIP*, vol. 50, pp. 261–273. Springer (2010)
15. Amálio, N., Kelsen, P., Ma, Q., Glodt, C.: Using VCL as an aspect-oriented approach to requirements modelling. TAOSD **VII**, 151–199 (2010)
16. Amálio, N., Payne, R., Cavalcanti, A., Woodcock, J.: Checking SysML models for co-simulation. In: ICFEM 2016, *LNCS*, vol. 10009. Springer (2016)
17. Amálio, N., Polack, F., Stepney, S.: An object-oriented structuring for Z based on views. In: ZB 2005, *LNCS*, vol. 3455, pp. 262–278 (2005)
18. Amálio, N., Polack, F., Stepney, S.: UML+Z: Augmenting UML with Z. In: H. Abrias, M. Frappier (eds.) Software Specification Methods, pp. 81–102. ISTE (2006)
19. Amálio, N., Stepney, S., Polack, F.: Formal proof from UML models. In: Proc. ICFEM 2004, *LNCS*, vol. 3308, pp. 418–433. Springer (2004)
20. Arisholm, E., Briand, L.C., Hove, S.E., Labiche, Y.: The impact of UML documentation on software maintenance: an experimental evaluation. IEEE TSE **32**(6), 365–381 (2006)
21. Arisholm, E., Sjøberg, D.I.: Evaluating the effect of a delegated versus centralized control style on the maintainability of object-oriented software. IEEE Transactions on Software Engineering **30**(8), 521 – 534 (2004)
22. Benjamini, Y., Hochberg, Y.: Controlling the false discovery rate: A practical and powerful approach to multiple testing. Journal of the Royal Statistical Society. Series B **57**(1), 289–300 (1995)
23. Benjamini, Y., Yekutieli, D.: The control of the false discovery rate in multiple testing under dependency. The Annals of Statistics **29**(4), 1165–1188 (2001)
24. Blackwell, A., Britton, C., Cox, A., Green, T., Gurr, C., Kadoda, G., Kutar, M., Loomes, M., Nehaniv, C., Petre, M., Roast, C., Roes, C., Wong, A., Young, R.: Cognitive dimensions of notations: Design tools for cognitive technology. Cognitive Technology pp. 325–341 (2001)
25. Blackwell, A.F., Whitley, K.N., Good, J., Petre, M.: Cognitive factors in programming with diagrams. Artificial Intelligence review **15**, 95–114 (2001)
26. Booch, G.: Object-Oriented Analysis and Design with applications. Addison-Wesley (1994)
27. Bottoni, P., Koch, M., Parisi-Presicce, F., Taentzer, G.: A visualisation of OCL using collaborations. In: UML 2001, *LNCS*, vol. 2185, pp. 257–271. Springer (2001)
28. Briand, L., Labiche, Y., Madrazo-Rivera, R.: An experimental evaluation of the impact of system sequence diagrams and system operation contracts on the quality of the domain model. In: Empirical Software Engineering and Measurement. IEEE (2011)
29. Briand, L.C., Labiche, Y., Penta, M.D., Yan-Bondoc, H.: An experimental investigation of formality in UML-based development. IEEE TSE **31**(10), 833–849 (2005)
30. Chen, P.C.H.: Why diagrams are (sometimes) six times easier than words: Benefits beyond locational indexing. In: Diagrams 2004 (2004)
31. Chen, P.P.S.: The entity-relationship model—toward a unified view of data. ACM Transactions on Database Systems **1**(1), 9–36 (1976)
32. Clark, T., Evans, A., Kent, S., Brodsky, S., Cook, S.: A feasibility study in rearchitecting uml as a family of languages using a precise oo meta-modeling approach. Tech. rep. (2000)
33. Cohen, J.: Statistical power analysis for the behavioural sciences. Lawrence Erlbaum Associates (1988)
34. Cohen, J.: The earth is round ($p < .05$). American Psychologist **49**(2), 997–1003 (1994)
35. Cook, S., Kleppe, A., Warmer, J., Mitchell, R., Rumpe, B., Wills, A.C.: Defining UML family members using prefaces. In: TOOLS '99 (1999)
36. Cumming, G.: Understanding the new statistics: Effect sizes, confidence intervals and meta-analysis. Routledge (2012)

37. Cumming, G.: The new statistics: Why and how. Psychological Science (2013)
38. Cumming, G., Finch, S.: 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)
39. Davis, F.D.: Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Quarterly **13**(3), 319–340 (1989)
40. Dobing, B., Parsons, J.: How UML is used. Comunications of the ACM **49**(5), 109–113 (2006)
41. Ehrig, K., Winkelmann, J.: Model transformation from visual OCL to OCL using graph transformation. ENTCS **152**, 23–37 (2006)
42. Evans, A., France, R., Lano, K., Rumpe, B.: The UML as a formal modeling notation. In: UML'98, *LNCS*, vol. 1618, pp. 336–348. Springer (1998)
43. Falessi, D., Juristo, N., Wohlin, C., Turhan, B., Münch, J., Jedlitschka, A., Oivo, M.: Empirical software engineering experts on the use of students and professionals in experiments. Empirical Software Engineering (2017)
44. Farias, K., Garcia, A., Lucena, C.: Evaluating the impact of aspects on inconsistency detection effort: a controlled experiment. In: Models 2012, *LNCS*, vol. 7590, pp. 219–234 (2012)
45. Farias, K., Garcia, A., Whittle, J.: Assessing the impact of aspects on model composition effort. In: ACM (ed.) AOSD 2010, pp. 73–84 (2012)
46. Ferguson, E.S.: The mind's eye: nonverbal thought in technology. Science **197**(4306) (1977)
47. Ferguson, E.S.: Engineering and the Mind's eye. MIT Press (1992)
48. Fish, A., Flower, J., Howse, J.: The semantics of augmented constraint diagrams. Journal of Visual Languages and Computing **16**, 541–573 (2005)
49. France, R., Ghosh, S., Dinh-Trong, T.: Model-driven development using UML 2.0: Promises and pitfalls. IEEE Computer **39**(2), 59–66 (2006)
50. Franz, V.H., Loftus, G.R.: 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)
51. Glass, R.L.: The software-research crisis. IEEE Software **11**(6), 42–47 (1994)
52. Golkasian, P.: Picture-word differences in a sentence verification task. Memory & cognition **24**(2), 584–594 (1996)
53. Goolkasian, P.: Pictures, words, and sounds: from which format are we best able to reason? Journal of General Psychology **127**(4), 439–459 (2000)
54. Green, T.R.G., Petre, M.: Usability analysis of visual programming environments: A 'cognitive dimensions' framework. Journal of Visual Languages and Computing **7**, 131—174 (1996)
55. Greenwald, A.G.: Within-subjects designs: To use or not to use? Psychological Bulletin **83**(2), 314–320 (1976)
56. Grissom, R.J., Kim, J.J.: Effect sizes for Research: A broad practical approach. Lawrence Erlbaum Associates (2005)
57. Harel, D.: Statecharts: a visual formalism for complex systems. Science of Computer Programming **8**, 231–274 (1987)
58. Harel, D.: On visual formalisms. Commun. of the ACM **31**(5), 514–530 (1988)
59. Harel, D.: Biting the silver bullet. IEEE Software **25**(1), 8 – 20 (1992)
60. Henderson-Sellers, B.: UML – the good, the bad or the ugly? perpectives from a panel of experts. Software and Systems Modeling **3**(1), 4–13 (2005)
61. Henderson-Sellers, B., Barbier, F.: Black and white diamonds. In: UML'99, *LNCS*, vol. 1723, pp. 550–565. Springer (1999)
62. Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics **6**(2), 65–70 (1979)
63. Höst, M., Regnell, B., Wohlin, C.: Using students as subjects – a comparative study of students and professionals in lead-time impact assessment. Empirical Software Engineering **5**(3), 201–214 (2000)
64. ISO: Information technology—Z formal specification notation—syntax, type system and semantics (2002). ISO/IEC 13568:2002, Int. Standard
65. Kelley, K.: Confidence intervals for standardized effect sizes: Theory, application, and implementation. Journal of Statistical Software **20**(8), 1–24 (2007)
66. Kelsen, P.: A declarative executable model for object-based systems based on functional decomposition. In: ICSOFT 2006, pp. 63–71 (2006)

67. Kelsen, P., Ma, Q.: A lightweight approach for defining the formal semantics of a modeling language. In: Models 2008, *LNCS*, vol. 5301, pp. 690–704 (2008)
68. Kim, Y.G., March, S.T.: Comparing data formalisms. CACM **38**(6), 103–115 (1995)
69. Lange, C.F., Chaudron, M.R.: Effects of defects in UML models – an experimental investigation. In: ICSE 2006, pp. 401–410. IEEE (2006)
70. Larkin, J.H., Simon, H.A.: Why a diagram is (sometimes) worth ten thousand words. Cognitive Science **11**, 65–99 (1987)
71. Larman, C.: Applying UML and patterns. Addison-Wesley (2004)
72. Leemans, J., Amálio, N.: Modelling a cardiac pacemaker visually and formally. In: VL/HCC 2012, pp. 257–258. IEEE (2012)
73. Leemans, J., Amálio, N.: A VCL model of a cardiac pacemaker. Tech. Rep. TR-LASSY-12-04, Univ. of Luxembourg (2012). `http://bit.ly/xiob5d`
74. Manfred Broy, M.V.C.: UML formal semantics: lessons learned. Software and Systems Modeling **10**(4), 441–446 (2011)
75. Meyer, B.: Applying "design by contract". Computer **25**(10), 40–51 (1992)
76. Micskei, Z., Waeselynck, H.: The many meanings of UML 2 sequence diagrams: a survey. Software and Systems Modeling **10**(4), 489–514 (2011)
77. Moody, D., van Hillegersberg, J.: Evaluating the visual syntax of UML: An analysis of the cognitive effectiveness of the UML family of diagrams. In: SLE, *LNCS*, vol. 5452. Springer (2009)
78. Moody, D.L.: Complexity effects on end user understanding of data models: an experimental comparison of large data model representation methods. In: ECIS 2002 (2002)
79. Moody, D.L.: The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering. IEEE TSE **6**(35), 756–779 (2009)
80. Mussbacher, G., Alam, O., Alhaj, M., Ali, S., Amálio, N., Barn, B., Bræk, R., Clark, T., Combemale, B., Cysneiros, L.M., Fatima, U., France, R., Georg, G., Horkoff, J., Kienzle, J., Leite, J.C., Lethbridge, T.C., Luckey, M., Moreira, A., Mutz, F., Oliveira, A.P.A., Petriu, D.C., Schöttle, M., Troup, L., Werneck, V.M.B.: Assessing composition in modeling approaches. In: CMA '12. ACM (2012)
81. Newcombe, R.G.: Two-sided confidence intervals for the single proportion: comparison of seven methods. Statistics in Medecine **17**(8), 857–72 (1998)
82. Newcombe, R.G., Altman, D.G.: Proportions and their differences. In: Statistics with Confidence: Confidence Intervals and Statistical Guidelines. Wiley (2000)
83. Nuzzo, R.: Scientific method: Statistical errors. Nature **506**, 150–152 (2014)
84. OMG: OMG systems modeling language, version 1.3. Tech. rep., OMG (2012)
85. Oppenheim, A.N.: Questionnaire Design, Interviewing and Attitude Measurement. Continuum (1996)
86. Petre, M.: Why looking isn't always seeing. Communications of the ACM **6**(38) (1995)
87. Pfister, R., Janczyk, M.: Confidence intervals for two sample means: Calculation, interpretation, and a few simple rules. Advances in cognitive Psychology (2013)
88. Pinker, S.: A theory of graph comprehension. In: R. Freedle (ed.) Artificial Inteligence and the future of testing, pp. 73–126 (1990)
89. Purchase, H.C., Colpoys, L., McGill, M., Carrington, D.: UML collaboration diagram syntax: an empirical study of comprehension. In: VISSOFT 2002, pp. 13–22. IEEE (2002)
90. Purchase, H.C., Colpoys, L., McGill, M., Carrington, D., Britton, C.: UML class diagram syntax: an empirical study of comprehension. In: APVis'01 (2001)
91. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2015). URL `https://www.R-project.org/`
92. Ricca, F., Penta, M.D., Torchiano, M., Tonella, P., Ceccato, M.: The role of experience and ability in comprehension tasks supported by uml stereotypes. In: ICSE 2007. IEEE (2007)
93. Ricca, F., Penta, M.D., Torchiano, M., Tonella, P., Ceccato, M.: How developers' experience and ability influence web application comprehension tasks supported by uml stereotypes: A series of four experiments. IEEE Transactions on Software Engineering **36**(1), 96–118 (2010)
94. Richters, M.: A precise approach to validating UML models and OCL constraints. Ph.D. thesis, Universität Bremen (2001)
95. Richters, M., Gogolla, M.: On formalizing the UML object constraint language OCL. In: ER '98, *LNCS*, vol. 1507, pp. 449–464. Springer (1998)

96. Ross, D.T.: Structured analysis (SA): A language for communicating ideas. IEEE Transactions on Software Engineering **3**(1), 16–34 (1977)
97. Ross, D.T., Schoman, K.E.: Structured analysis for requirements definition. IEEE Transactions on Software Engineering **3**(1), 6–15 (1977)
98. Rumbaugh, J., Blaha, M., Permerlani, W., Eddy, F., Lorensen, W.: Object Oriented Modelling and Design. Prentice-Hall (1991)
99. Rumpe, B., France, R.: Variability in UML language and semantics. Software and Systems Modeling **10**(4), 439–440 (2011)
100. Salman, I., Misirli, A.T., Juristo, N.: Are students representatives of professionals in software engineering experiments? ICSE'15 pp. 666–676 (2015)
101. Shimojima, A.: Operational constraints in diagrammatic reasoning. In: G. Allwein, J. Barwise (eds.) Logical Reasoning with Diagrams (1996)
102. Sjøberg, D.I.K., Anda, B., Arisholm, E., Dyba, T., Jorgensen, M., Karahasanovic, A., Koren, E., Vokac, M.: Conducting realistic experiments in software engineering. In: International Symposium on Empirical Software Engineering (ISESE '02) (2002)
103. Spivey, J.M.: The Z Notation: A Reference Manual, 2nd edn. Prentice Hall (1992)
104. Staron, M., Kuzniarz, L., Wohlin, C.: Empirical assessment of using stereotypes to improve comprehension of uml models: A set of experiments. Journal of Systems and Software **79**(5), 727–742 (2006)
105. Stevens, P.: On the interpretation of binary associations in the unified modelling language. Software and Systems Modeling **1**(1), 68–79 (2002)
106. Störrle, H.: Semantics of interactions in UML 2.0. In: Human Centric Computing Languages and Environments (HCC 2003), pp. 129–136. IEEE (2003)
107. Tobias, E., Ras, E., Amálio, N.: Suitability of visual modelling languages for modelling tangible user interface applications. In: VL/HCC 2012, pp. 269–270. IEEE (2012)
108. Torchiano, M.: Empirical assessment of UML static object diagrams. In: Program Comprehension, 2004. IEEE (2004)
109. Varró, D.: A formal semantics of UML statecharts by model transition systems. In: ICGT 2002, LNCS, pp. 378–392. Springer (2002)
110. Wieringa, R.: A survey of structured and object-oriented software specification methods and techniques. ACM Computing Surveys **30**(4), 459–527 (1998)
111. Wilcox, R.R., Keselman, H.J.: Modern robust data analysis methods: measures of central tendency. Psychological Methods **8**(3), 254–74 (2003)
112. Wilcox, R.R., Tian, T.S.: Measuring effect size: a robust heteroscedastic approach for two or more groups. Journal of Applied Statistics **38**(7), 1359–1368 (2011)
113. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer (2012)
114. Woodcock, J., Davies, J.: Using Z: Specification, Refinement, and Proof. Prentice-Hall (1996)
115. Yuen, K.K.: The two-sample trimmed t for unequal population variances. Biometrika **61**(1), 65–170 (1974)
116. Yusuf, S., Kagdi, H., Maletic, J.I.: Assessing the comprehension of UML class diagrams via eye tracking. In: Program Comprehension, 2007. IEEE (2007)
117. Zhang, J.: The nature of external representations in problem solving. Cognitive Science **21**(2), 179–217 (1997)