Preliminary Validation of a Lightweight Approach to Consistency of Scenarios and Class Models

Martin Glinz Nancy Schett

Technical Report 2000.02

Institut für Informatik, Universität Zürich
Winterthurerstrasse 190
CH-8057 Zurich, Switzerland
+41-1-63 54570
http://www.ifi.unizh.ch/~glinz

Abstract

In [2] we present a lightweight approach to consistency between a scenario model and a class model that is based on minimizing overlap between the models and on systematic cross-referencing.

In this paper, we describe a preliminary experimental validation of our approach. The results clearly indicate that the participants in the experiment preferred our lightweight approach to a classical approach.

1 Introduction

Today, most approaches to object-oriented requirements specification use a combination of structure, behavior and interaction models for (functional) requirements specification. Typically, structure and behavior are represented in class models that consist of a combination of class diagrams and statecharts, whereas interaction is modeled with use cases / scenarios [1], [3].

As soon as more than one model is used, the problem of inter-model consistency arises: how can we ensure that information in these models is neither contradictory nor partially incomplete? (A partial incompleteness is a situation where information given in one model requires corresponding information in another model which, however, is missing from that model.)

Nearly all requirements modeling techniques that use more than one model have *no systematic approach to combining the models consistently*. The consistency problem is simply ignored (and thus left to the requirements engineers and to the users who have to validate a requirements specification).

In [2], we investigate the consistency problem between a scenario model (or use case model) and a class model and introduce a concept for making them consistent. We use a lightweight approach for loosely coupled, UML-style models. Our approach is lightweight in the sense that we use semi-formal models where consistency cannot be established formally.

The basic idea of our approach is to achieve consistency by *minimizing overlap* between the two models and by *systematically cross-referencing* corresponding information. We introduce two cross-referencing schemes, a simple one and an elaborate one. We give a set of rules that can be used both for developing a consistent specification and for checking the consistency of a completed specification. Some rules can be checked automatically, the others are rules for manual inspection.

In this paper, we describe a small experiment we conducted with graduate and PhD students in order to validate our approach.

2 Goal of the experiment

The goal of this experiment was to compare models with and without cross-references with respect to their acceptance both for writing consistent specifications and for consistency checking.

3 Setup of the experiment

We used the department library system given in [2] as an example (see below).

Sample Application: The Department Library System

Goal: The system shall support a department library, where students themselves can take books from the shelves, read them, borrow or return books, and query the library catalog. Self-service terminals shall be used for borrowing and returning books.

Constraints: Users identify themselves with a personal library card. Every book has a barcode label. Additionally, there is an electronic safety device located under that barcode label which can be turned on and off by the system. At the exit of the library, a security gate sounds an alarm when a user tries to leave the library with a non-checked-out book. A user may not borrow a book if she has currently borrowed books that are overdue. A book may not be borrowed if it is reserved for another user.

Based on this example, we set up three specifications of the same requirements.

- **Specification A:** a conventional UML model collection.
- **Specification B:** the simple scheme of cross-referencing (as defined in [2], Section 3.4)

- **Specification C:** the elaborate scheme of cross-referencing as defined in [2], Section 3.5).

The three specifications are given in Appendix 1.

Twelve students participated in the experiment. They first received the three specifications and had to answer some questions about each of them (to make sure that they thoroughly studied and understood the models). For each model they were given 20 minutes time to study the model and to answer the questions (Questionnaire I in Appendix 2). Then they had to answer questions about their preferences concerning the three types of specifications. For this questionnaire (Questionnaire II in Appendix 2) they had 10 minutes time. As all participants were German speaking, we gave them the questionnaires in German (see Appendix 2). A translation of Questionnaire II into English is given in Appendix 3.

4 Threats to the validity of the experiment

4.1 Sequence bias

In order to exclude bias due to the sequence of working through the tree specifications, we divided the participants into four groups that did the specification in different order:

- Group 1: A B C
- Group 2: C A B
- Group 3: B A C
- Group 4: C B A.

4.2 Number of participants

The fact that we had only 12 participants is the weakest point in this experiment. This number is too small for calculating percentages and statistical significance. So we can only deduct trends from the results but no statistically significant quantitative values.

4.3 Kind of participants

As the participants were students, the experiment is not representative for industrial practice. Practitioners might have different needs and perceptions concerning consistency of models.

5 Results

The evaluation of Questionnaire II is given in Figures 1 to 4. Figure 1 shows the overall preference of the participants. Figures 2 and three show how the participants rated the three specifications with respect to their suitability for writing a consistent specification (Figure 2) and for checking an existing specification for consistency (Figure 3). Finally, Figure 4 shows how the participants rated the overall benefit/cost ratio of the three approaches.

Due to the small number of participants, we do not calculate statistics. However, the figures show a very clear preference for specifications using either the simple or the elaborate cross-referencing scheme.

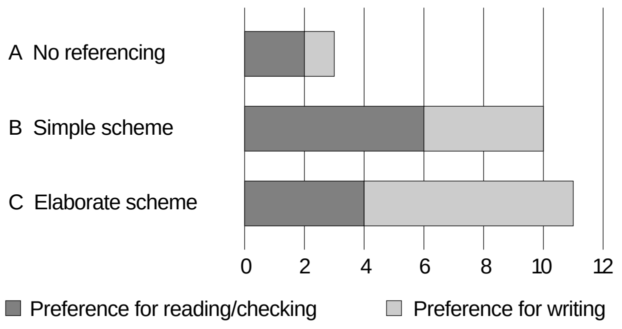


Figure 1. Overall preferences of participants

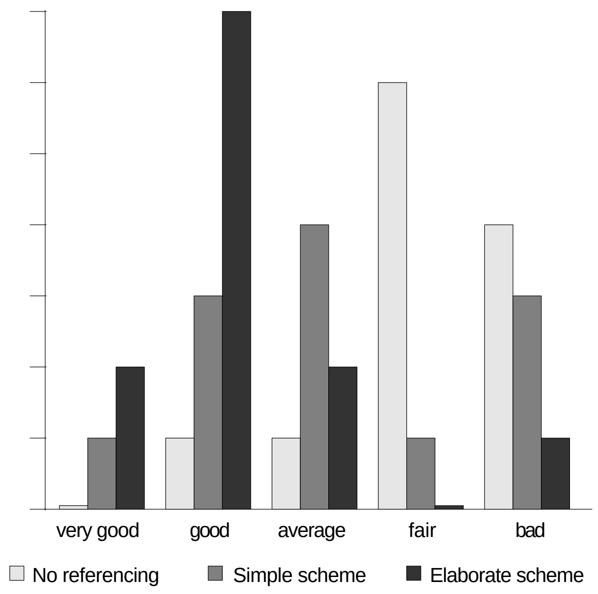


Figure 2. Suitability for writing a consistent specification

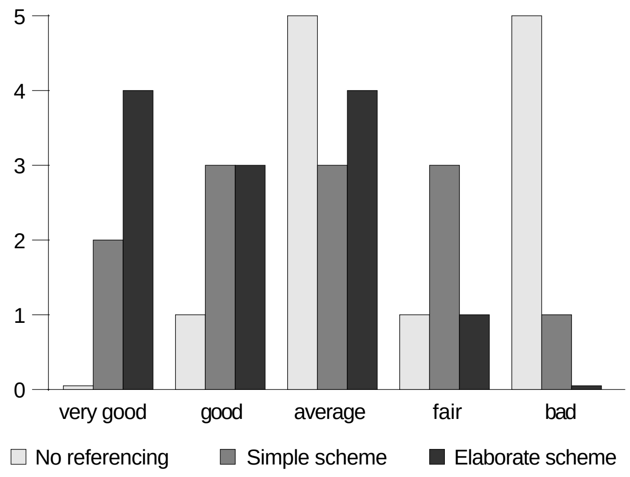


Figure 3. Suitability for checking consistency[1]

[1] Unfortunately, we found two counting errors in the first evaluation of the questionnaires when we cross-checked the data while preparing the final version of this report. As a consequence, Figure 9 in [2] is slightly incorrect: the correct results for the simple scheme are good: 3, bad: 1, while Figure 9 in [2] reports good: 2, bad: 2. For the elaborate scheme, the correct figures are very good: 4, good: 3, while Figure 9 in [2] reports very good: 3, good: 4. However, both errors do not falsify our conclusions because the correct data are even more in favor of our approach than the data given in [2].

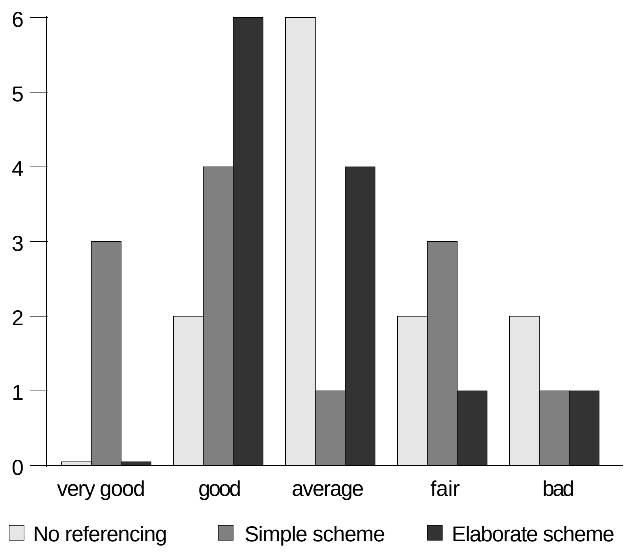


Figure 4. Benefit/cost ratio

6 Conclusions

As already mentioned, the small number of participants does not allow us calculating statistically significant values about the differences between the three approaches. However, all four figures clearly show the same trend: the simple and the elaborate scheme of cross-referencing are rated considerably better than the classic approach with no referencing.

The elaborate scheme is rated higher than the simple scheme concerning the suitability both for writing a consistent specification and for checking a specification for consistency. In the overall preference, there is not much difference between the two approaches. We suspect that the relatively high rating of the elaborate scheme (specification variant C) is due to the fact that students prefer precise, programming language-like approaches. Practitioners presumably would rate the simple scheme higher than the elaborate one.

The high rating of the benefit/cost ratio for the simple scheme is remarkable.

Summing up we can state that this experiment has been a small and preliminary one and hence we can draw limited conclusions only. However, we have a clear indication that the experimental results support our approach and that it is worthwhile to continue our work on a lightweight approach to inter-model consistency.

References

[1] Firesmith, D., Henderson-Sellers, B. H., Graham, I., Page-Jones, M. (1998). *Open Modeling Language (OML) – Reference Manual.* SIGS reference library series. Cambridge, etc.: Cambridge University Press.

[2] Glinz, M. (2000). *A Lightweight Approach to Consistency of Scenarios and Class Models.* To appear in Proceedings of the IEEE International Conference on Requirements Engineering. http://www.ifi.unizh.ch/groups/req/ftp/papers/ICRE2000.pdf

[3] Rumbaugh, J., Jacobson, I., Booch, G. (1999). *The Unified Modeling Language Reference Manual.* Reading, Mass., etc.: Addison-Wesley.

Appendix 1: Specifications A, B, C

Variant A: Classic UML Model

1. Scenario/use case Borrow Books (Variant A)

Use Case:	Borrow books
Actor:	User
Precondition:	User has personal library card
Started by:	User wants to borrow books

Normal flow:

1 User scans her library card
System validates the card, returns the card, displays user data, displays ‘Select function’ dialog

2 User selects ‘Borrow’ function
System displays ‘Borrow’ dialog

3 User scans label of book to be borrowed
System identifies book, creates an ‘on-loan’ record (consisting of book, user and current date), unlocks safety label, displays book data

4 If user presses ‘More books’ key,
System displays ‘Borrow’ dialog. Repeat step 3

5 User presses ‘Finish’ key
System prints loan slip, displays ‘Finished’ message

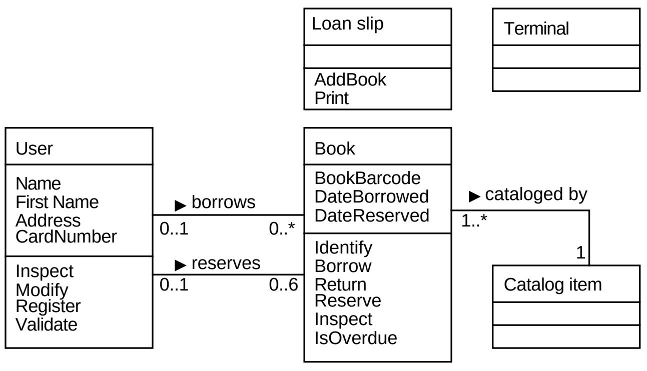
Alternative flows:

1a Card is invalid: System returns the card, displays ‘Invalid’ message (scenario terminates)

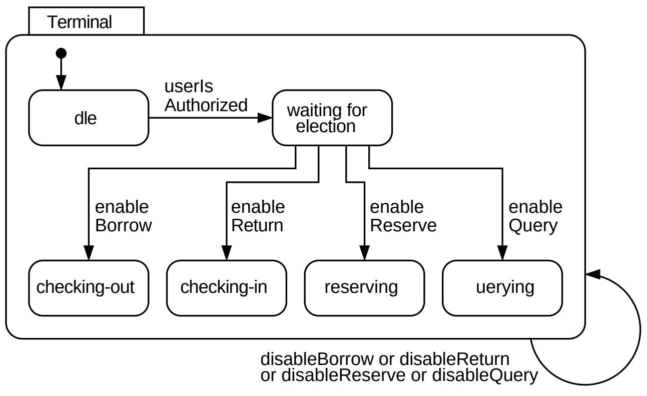
2a User has overdue books: System displays ‘Denied’ message (scenario terminates)

3a Book is reserved for another person: System displays ‘Reserved’ dialog (scenario continues)

2. Class model of the library (Variants A and B)



3. Behavior model for objects of class Terminal



Variant B: UML Model with cross-references in the scenarios/use cases to other elements of the model

1. Scenario/use case Borrow Books (Variant B)

Use Case:	Borrow books
Actor:	User
Precondition:	User has personal library card
Started by:	User wants to borrow books

Normal flow:

1 User scans her library card
System validates the card (↑User.Validate), returns the card, displays user data, enables selection (↑Terminal. userIsAuthorized), displays ‘Select function’ dialog

2 User selects ‘Borrow’ function
System enables borrowing (↑Terminal.enableBorrow), displays ‘Borrow’ dialog

3 User scans label of book to be borrowed
System identifies book (↑Book.Identify), records the book as borrowed (↑Book.Borrow), unlocks safety label, displays book data

4a User presses ‘More books’ key
System displays ‘Borrow’ dialog. Repeat step 3

4b User presses ‘Finish’ key
System disables borrowing (↑Terminal.disableBorrow), prints loan slip (↑LoanSlip.Print), displays ‘Finished’ message

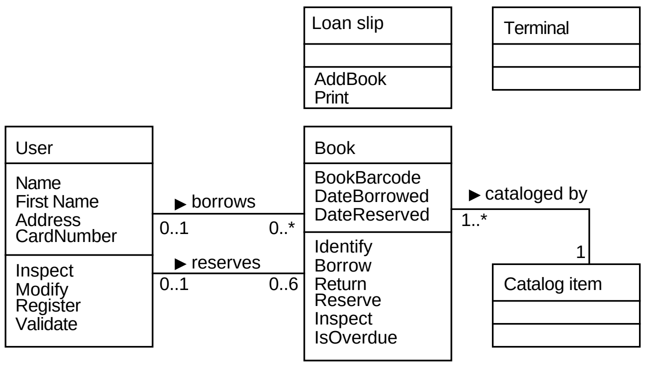
Alternative flows

1’ If Card is invalid (determined by ↑User.Validate): System returns the card, displays ‘Invalid’ message (scenario terminates)

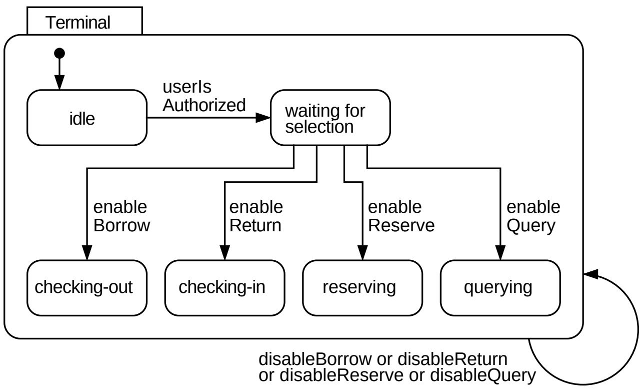
2’ If User has overdue books (determined by ↑User.Validate): System displays ‘Denied’ message (scenario terminates)

3’ If Book is reserved for another person (determined by ↑Book.Borrow): System displays ‘Reserved’ dialog (scenario continues)

2. Class model of the library (Variants A and B)



3. Behavior model for objects of class **Terminal**



Variant C: Extended UML Model with elaborate cross-references between scenarios/use cases and class model

1. Scenario/use case **Borrow Books** (Variant C)

Use Case:	Borrow books
Actor:	User
Precondition:	User has personal library card
Started by:	User wants to borrow books

Normal flow:

1 User scans her library card {**delivers** numberOfCard}
System validates the card {↑User.Validate (**in** numberOfCard, **out** result, **out** currentUser, **out** userData); return card; **check step** 1'; display userData; ↑Terminal.userIsAuthorized; display 'Select function' dialog}

2 User selects 'Borrow' function
System begins check-out session {**check step** 2'; ↑Terminal. enableBorrow; display 'Borrow' dialog}

3 User scans label of book to be borrowed {**delivers** label}
System checks-out book {↑Book.Identify (**in** label, **out** theBook); ↑theBook.Borrow (**in** currentUser, **out** bookData, **out** status); **check step** 3'; unlock safety label; display bookData}

4a User presses 'More books' key
System iterates {display 'Borrow' dialog; **go to step** 3}

4b User presses 'Finish' key
System terminates check-out session {↑Terminal.disable-Borrow; ↑currentLoanSlip.Print; display 'Finished' message; **terminate**}

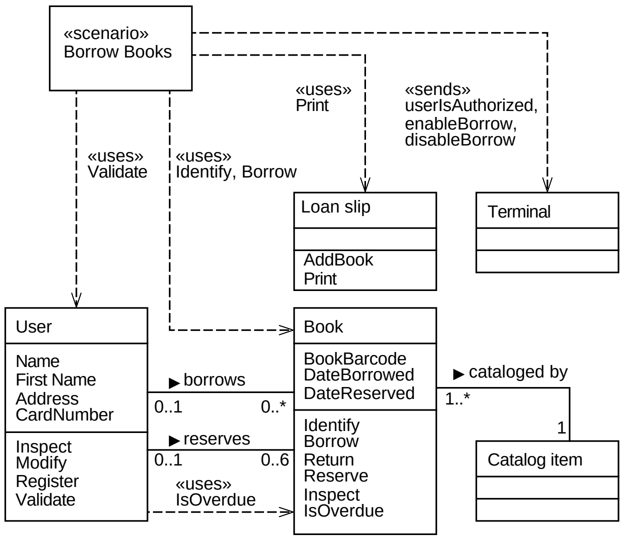
Alternative flows:

1' Card is invalid {**if** (result = "invalid card") display 'Invalid' message; **terminate**; **endif**}

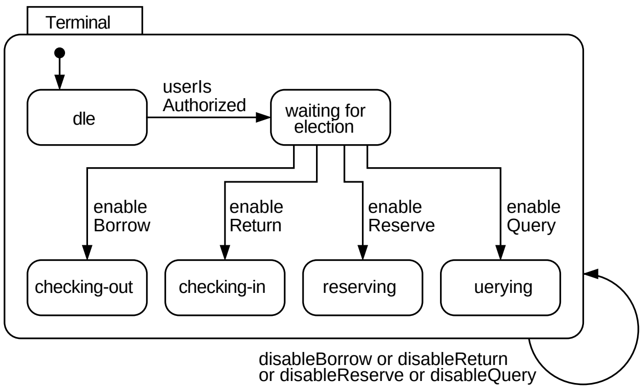
2' User has overdue books {**if** (result = "valid user with overdue books") display 'Denied' message; **terminate**; **endif**}

3' Book is reserved for another person {**if** (status = "reserved") display 'Reserved' dialog; **go to step** 4; **endif**}

2. Class model of the library (VariantC)



3. Behavior model for objects of class **Terminal**



4. Specification of the operation **Validate** in class **User** (Variant C)

```
class operation Validate (in numberOfCard: Number, out result:
                ValidationResult, out id: User, out userData: String)
    in class User
pre   Terminal is in state "idle"
post  if (exists x in User • x.CardNumber = numberOfCard)
        if (for all b in x.borrows • b.IsOverdue = false)
          result = "valid user"; id = x, userData = (x.FirstName,
          x.Name, x.CardNumber)       // x.borrows is the set of all
                                      // books that have been
                                      // borrowed by user x
        else result = "valid user with overdue books"
        endif
      else result = "invalid card"
      endif
end
```

Appendix 2: Questionnaires (in German)

I. Sachfragen

Modellvariante: **Versuchsperson Nr.**

Frage	**Aufwand für Antwort**		
	gering	mittel	hoch
1. Wo sind die Anforderungen, wie eine Karte zu validieren ist, modelliert?	❏	❏	❏
2. Sind die modellierten Anforderungen zur Validierung der Karte ausreichend für eine Implementierung? ❏ ja ❏ weitgehend ja ❏ weitgehend nein ❏ nein	❏	❏	❏
3. Ist die Spezifikation eines ausgeliehenen Buchs im Szenario konsistent mit der Spezifikation im Klassenmodell? ❏ ja ❏ weitgehend ja ❏ weitgehend nein ❏ nein	❏	❏	❏
4. Wo sind die Ereignisse, welche das Verhalten eines Terminals steuern, modelliert?	❏	❏	❏
5. Ist die Spezifikation des Ausleihens eines Buchs vollständig? ❏ ja ❏ nein Wenn nein: Was fehlt?	❏	❏	❏

II. Beurteilungsfragen (gemeinsam für alle Varianten)

Versuchsperson Nr.

Frage		**Variante**	
	A	**B**	**C**
1. Welche der drei Varianten würden Sie persönlich bevorzugen			
a) als Leser/Prüfer einer Spezifikation?	❑	❑	❑
b) als Requirements Ingenieur, der eine Spezifikation schreibt?	❑	❑	❑

Frage		**sehr gut**	**gut**	**befriedigend**	**ausreichend**	**schlecht**
2. Wie beurteilen Sie die drei Varianten in Bezug auf ihre Eignung						
a) eine konsistente Spezifikation zu schreiben?	**A**	❑	❑	❑	❑	❑
	B	❑	❑	❑	❑	❑
	C	❑	❑	❑	❑	❑
b) eine Spezifikation auf Konsistenz zu prüfen?	**A**	❑	❑	❑	❑	❑
	B	❑	❑	❑	❑	❑
	C	❑	❑	❑	❑	❑
3. Wie beurteilen Sie die drei Varianten in Bezug auf ihren Nutzen im Vergleich zum Aufwand?						
	A	❑	❑	❑	❑	❑
	B	❑	❑	❑	❑	❑
	C	❑	❑	❑	❑	❑

Zur Erinnerung: Die drei Varianten sind

A	klassisch, ohne Querverweise
B	mit einfachen Querverweisen im Szenario/Anwendungsfall
C	mit ausführlichen, gegenseitigen Querverweisen

Appendix 3: English translation of Questionnaire II

II. Rating Questions

Participant No.

(Jointly for all three variants)

Question		**Variant** A	B	C
1.	Which of the three variants would you personally prefer			
a)	as a reader/checker of a specification?	❏	❏	❏
b)	as a requirements engineer who is writing a specification?	❏	❏	❏

Question			**very good**	**good**	**average**	**fair**	**bad**
2.	How do you rate the three variants with respect to their suitability for						
a)	writing a consistent specification?	**A**	❏	❏	❏	❏	❏
		B	❏	❏	❏	❏	❏
		C	❏	❏	❏	❏	❏
b)	checking a specification for consistency?	**A**	❏	❏	❏	❏	❏
		B	❏	❏	❏	❏	❏
		C	❏	❏	❏	❏	❏
3.	How do you rate the three variants with respect to their benefit/cost ratio?						
		A	❏	❏	❏	❏	❏
		B	❏	❏	❏	❏	❏
		C	❏	❏	❏	❏	❏

Remember: the variants are

A	classic, no references
B	with simple references in the scenarios/use cases
C	with elaborate references