

Classes or Objects?

CRC-cards Considered Harmful

Extended Abstract

Jürgen Börstler
Umeå University, Sweden
jubo@cs.umu.se

1 Introduction

CRC-cards have originally been described as a tool for teaching object-oriented thinking to programmers (Beck & Cunningham, 1989). They are a simple informal tool for collaborative object-oriented modelling.

Many educators have adopted CRC-cards as a tool for teaching design early and a means to convey an object-oriented way of thinking (Biddle et al., 2002; Börstler et al., 2002).

The power of the CRC-card approach lies in its associated role-play activities. Scenario role-play is usually used in the social (behavioural) sciences to evaluate human behaviour in specific yet hypothetical situations. The role-play participants are assigned roles they enact according to predefined scenarios, much like actors following a script when playing the characters in play. During the role-play, the participants learn a lot about themselves, the other participants, and the roles played. The power of the role-play lies in its interactivity that supports creativity and sharing of knowledge. Role-playing is an effective way to simulate or explore hypothetical situations, since the characters and scripts (scenarios) can be easily varied.

In object-oriented software development, the characters are the objects in our system and the scenarios are situations of system usage. Scenario role-play can be used in very early stages of software development, before any code is written (or even designed). That makes it appealing for novices as a means to familiarise with the object-oriented way of thinking, without needing to bother with syntactic details of actual programming or design languages.

Almost all publications about the usage of CRC-cards and scenario role-play are quite positive and students seem to have little problems with the approach. However, the problems start when going into the details of the role-play activities. Our CS students often focus on implementation details and are generally not so good at abstraction. Furthermore, they often skip over details that would get them into interesting and/or important discussions about responsibility distribution. Many CRC texts explicitly recommend not to go into details. However, it is our firm belief that such discussions are necessary to really understand the importance of responsibility distribution (and our experience supports that belief).

2 CRC-cards are Objects and Classes

The CRC approach blurs the difference between objects and classes¹. In virtually all CRC texts the authors talk about finding candidate objects, like *book* or *account*. Then CRC-cards

¹ Some authors do this deliberately, like Beck and Cunningham (1989) in their original CRC paper.

(classes) are defined for significant candidates². Now the texts continue to talk about objects and classes, sometimes about instances and classes. That looks like straightforward OO terminology. However, when examining many original example candidate objects a little closer, like *book* and *account* we have a problem. The terms *book* or *account* are not instances they are only placeholders for instances, i.e. variables or "generic" objects, i.e. we had classes from the very beginning.

The problem is that the term object had different semantics during the pre-CRC phase (finding candidates) and the post-CRC phase (initial cards are available). Wirfs-Brock and McKean (2003) try to clean up that mess by introducing the term "role" for analysis and reserving "class"/"interface" for design. However, candidate objects are considered to be analysis "items". A role can be a single item or a group, but an object is always a single item. When documenting analysis models with CRC-cards (i.e. classes), the roles become classes, but we are still in the analysis phase ... I guess you get the point.

I think there is only one solution to this problem. **Do not use the term object without qualification in the pre-CRC phase.** Object (without qualification) should always mean the same thing, i.e. instance of an explicit or implicit class. Since there are no (explicit) classes in the pre-CRC phase, we must be specifically careful. *1984* for example can be validly interpreted as an object (of the implicit class `Book` for example), but talking about *book* as a candidate object is a misnomer that leads to quite some confusion for novices³. We propose to **ban the term candidate object completely.** Actually, **we only handle candidate classes** and we will introduce classes (CRC-cards) even if we only have single instances (a kind of `Singleton`)⁴.

Unfortunately, the confusion does not end with the creation of CRC-cards. During the role-playing of scenarios, most authors use the CRC-cards as object surrogates. "You can pick up a card and talk about it as if it were the object itself, forgetting that the card "stands in" for a "real" object." (Wirfs-Brock and Kean, 2003, p 99). However, this is technically not true. Different objects will most likely have different "actual knowledge".

Biddle et al (2002) describe similar problems. Interestingly they solved it the other way around. Their "approach was to increasingly say that the cards represented objects, not classes ... [and] found this helped in later roleplay". We suggest to make a clear distinction between objects and classes from the very beginning.

3 Further Issues and Problems

- **How do we handle user interface issues and interfaces to other systems?**
- **Which object will start a scenario?**
- **How do we know whether two objects can communicate?**
- **Which objects are known when a scenario starts and which ones need to be made accessible/ created first?**

² Kent Beck (1993, p 42) writes "The first C [of CRC card] stands for class, the name of the object." This must certainly be confusing for novices.

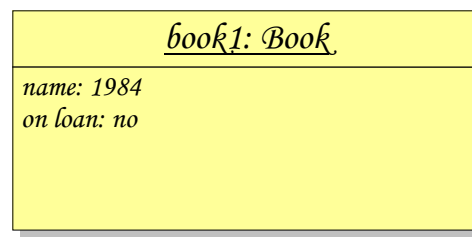
³ Beck and Cunningham (1989) explicitly deny that object/class confusion is a problem, but they have developed the approach for transitioning to OO and not for complete novices.

⁴ Please note that this thread of reasoning is invalid for classless OO languages, but in this case the original problem might not be an issue at all.

- How do we avoid getting lost in a scenario?
- How do we avoid implementation discussions?
- How detailed (complete) do we need to be in the role-play?
- How do we document scenarios?

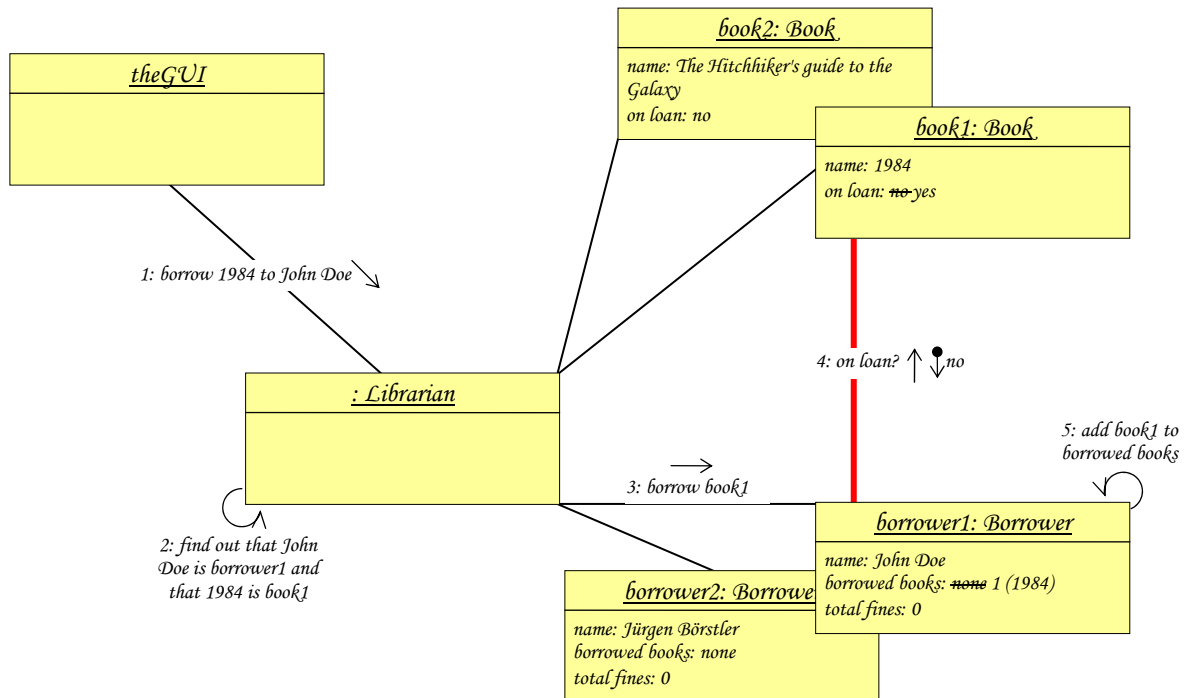
4 A Solution?

We propose the usage of *role-play diagrams* (RPDs) to support the role-play activity (Börstler, 2004). RPDs are semi-formal diagrams quite similar to UML collaboration/communication diagrams (Booch et al, 1999). RPDs are built successively as the scenario unfolds. The objects in RPDs, we call them object cards, are based on the UML-notation for objects. An object card is an instance of a CRC-card that shows the instance's name, class name and current knowledge, according to the information noted on its corresponding CRC-card. The role-play is acted out based on object cards. Large post-it notes work fine for this purpose. For each new object, we create a new object card and put it on the white-board or a large sheet of paper. The object card is initialised with the knowledge for this specific instance, using the corresponding CRC-card as a template. In a library system, we might for example have the following object card



Objects that "know" each other are connected by a line. During the role-play, communication is only possible between connected object cards (a GUI object is always available!). Each request (message send) is documented on the connecting line between the communicating objects by number: request. Number keeps track of the ordering of requests and request corresponds to the actual service requested (which need to be mapped to a responsibility). A small arrow denotes the direction of the request. Requests can be annotated with data flow for example to document information that is returned. When the knowledge of an object changes, the corresponding information on the object card is updated. At the end of a scenario role-play, we have an exact documentation of what happened, including the state changes. If necessary, this information could be translated to for example UML sequence diagrams.

In the example below, we show the resulting RPD after the scenario *John Doe borrows 1984*. Please note that certain preconditions for this scenario are included in the diagram, like for example whether the borrower already has borrowed books or outstanding fines or whether the book is on loan. Please note furthermore that the connecting line between `book1` and `borrower1` is not available at scenario start. It is drawn after request 3, i.e. when the librarian object has identified the actual objects and can tell the `borrower1` object to which book object it has to connect. In the same way, we document changes to the knowledge of objects.



The scenario above is somewhat simplified and does not take care of for example the computation of return dates or how exactly the borrowers keep track of borrowed books.

Our experiences so far are quite positive. Teachers as well as TAs think our approach to teaching CRC-cards has improved after the introduction of RPDs. Students are usually quite positive to the CRC-approach.

References

- Beck, K. (1993). CRC: Finding objects the easy way. *Object Magazine* 3 (4). 42-44.
- Beck, K., Cunningham, W. (1989). A Laboratory for Teaching Object-Oriented Thinking. *Proceedings OOPSLA '89*. 1-6.
- Biddle, R., Noble, J., Tempero, E. (2002). Reflections on CRC Cards and OO Design. *Proceedings TOOLS Pacific 2002*, Sydney, Australia.
- Booch, G., Rumbaugh, J., Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Reading, MA: Addison-Wesley.
- Börstler, J. (2004). Object-Oriented Analysis and Design Through Scenario Role-Play, Technical Report UMINF-04.04, Department of Computing Science, Umeå University, Sweden.
- Börstler, J., Johansson, T., Nordström, M. (2002). Introducing OO Concepts with CRC Cards and BlueJ—A Case Study, *Proceedings FIE'02*, Boston, USA, T2G-1-T2G-6.
- Wirfs-Brock, R., McKean, A. (2003). *Object Design--Roles, Responsibilities, and Collaborations*. Upper Saddle River, NJ: Prentice Hall.