The EasyCRC Tool

Draft version

1. Introduction

This paper presents the results of an effort to automate the process of defining classes from a written, "plain language" description of a system via CRC cards. We have created a tool to analyze text, to help find objects within it, to create CRC cards from the objects, and to help find the responsibilities of each of the items through sequence diagrams.
We have identified the need for a good tool to help students in OOSE courses. During OOSE courses, students are exposed to a plethora of new ideas and frequently have difficulty preparing assignments: identifying the cards from the requirements text, describing the scenarios simulations as use cases, and identifying the responsibilities and collaborators. This made the need for an efficient automating tool apparent.
Associating common nouns with data types makes the notion of data types more intuitive. Most people possess an intuitive understanding of common nouns [11].
We reviewed different tools that have the capability of editing CRC cards and creating other diagrams. We concluded that of the available tools that exist today, none enable the user to follow a productive CRC card method that will take the user all the way from a written "plain language" description to a complete set of CRC cards capable of being transformed into the appropriate classes. In most tools, the CRC card diagram seems to be detached from the other diagrams.
There is a need for a course design tool, specifically for students of OOSE, which can be used in the early stages of the design process. The tool must be easy to use, intuitive, to ensure that students' time is optimized in learning OOSE design, not spent on learning how to use the tool. In addition, the tool must be able to export its product to an XML format, allowing for interoperability with other tools.

1.1. About CRC Cards

1.1.1. Conceptual Overview

A CRC card is an index card used to represent the responsibilities of classes and the interaction between classes. CRC cards are an informal approach to object oriented modeling. The cards are created through scenarios, which model the behavior of the system based on the system requirements.

1.1.1.1. Why Use CRC Cards?

"CRC cards are portable; no computer is required for anytime, anywhere use.
This allows participants to experience how the system works firsthand. No software tool can replace the interaction that happens by physically picking up the cards and playing the role of that object.
They are a useful tool for teaching people the object-oriented paradigm. They can be used as a methodology themselves or as a front end to a more formal methodology such as Booch, Wirfs-Brock, Jacobson, etc.
Although CRC cards have mainly been invented for teaching, they have proven useful for much more. They became an accepted method for analysis and design. The biggest factor in their success is they provide an informal and non-threatening environment that is productive to working and learning." [3]

CRC cards have been adopted by many educators to teach early design in their object-oriented programming courses. [1]

Though CRC cards indeed can be used manually, we believe that a computer program could assist the CRC card user. When a complex system is analyzed, it becomes increasingly difficult to track all the data gathered and its context. A computerized tool could check and prevent different types of design errors, as we will show in chapter 3. Such a tool could facilitate updating all the cards when adding subclasses to a class (card) or deleting a class (card); tasks which are daunting and ridden with failure-points when done manually.

1.1.2. Introduction to CRC Cards and Role Play

A CRC card corresponds to a class. It describes the properties that certain types of objects of interest in the problem/application domain have in common. A class should have a single and well-defined purpose that can be described briefly and clearly. It should be named by a noun, noun phrase, or adjective that adequately describes the abstraction. The class name is written across the top of the CRC-card. A short description of the purpose of the class is written on the back of the card.

A *responsibility* is something the objects of a class take care of; a service provided for other objects. A responsibility can be either to know something or to do something. A book object in a library, for example, might (among other things), be responsible for checking itself out, knowing its title and knowing whether it is on loan. To do something, an object usually uses its (local) knowledge. If this knowledge is not sufficient for the purpose, the object can require help from other objects (its collaborators; see below). The responsibilities of a class are written along the left side of the card.

The *collaborators* describe which kinds of objects can be asked for help to fulfill a specific responsibility. An object of a collaborator class can, for example, provide further information, or actually take over parts of the original responsibility (by means of the

collaborator's own responsibilities). A book object for example can only know whether it is overdue if it also knows the current date. As shown in Figure 1, this information is provided by the collaborator date. Collaborators are written along the right side of the card in the same row as the corresponding responsibility.

The back of the card can be used for the class' description, comments and miscellaneous details." [1]

Sequence diagrams are effective alternatives for describing a role-playing session. The parts of the use case are essentially the subtasks considered in the CRC card session; and the labels on the interaction arrows are the responsibilities assigned to that object to carry out various sub-tasks [4, page 97].

The rest of this paper is organized a follows:
Chapter 2 - reviews existing CRC cards edit tools.
Chapter 3 - presents our tool and describes how to use it.
Chapter 4 -presents our conclusions.

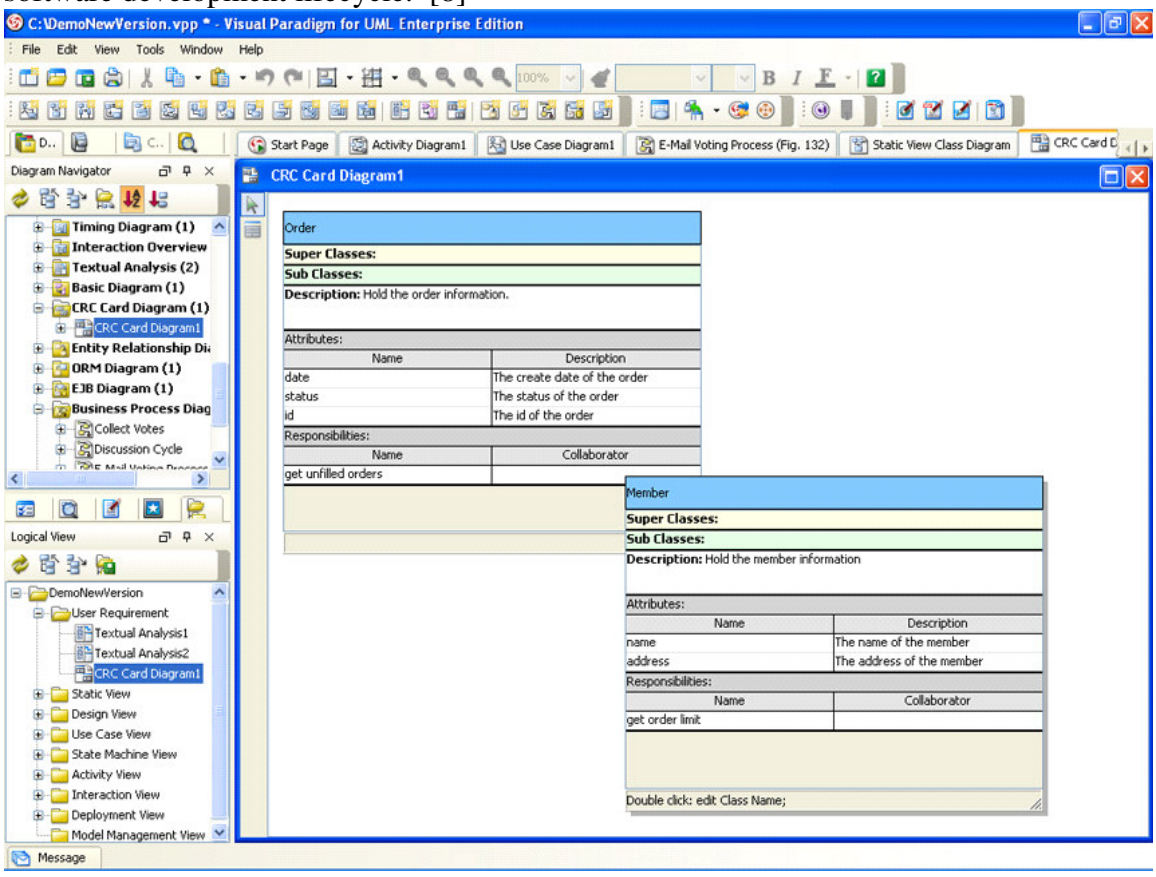
2. Related work

We have tried to review as many tools as were available through the internet and other resources. In this section, we will present only tools with the capabilities to design and edit CRC cards. The tools we found can be divided into two major groups – commercial tools and free\academic tools. Not all the tools were tested, because some of them were not available. The software versions we used were free or the trial versions; this limited our observations.

2.1. Commercial Tools

2.1.1. Visual Paradigm

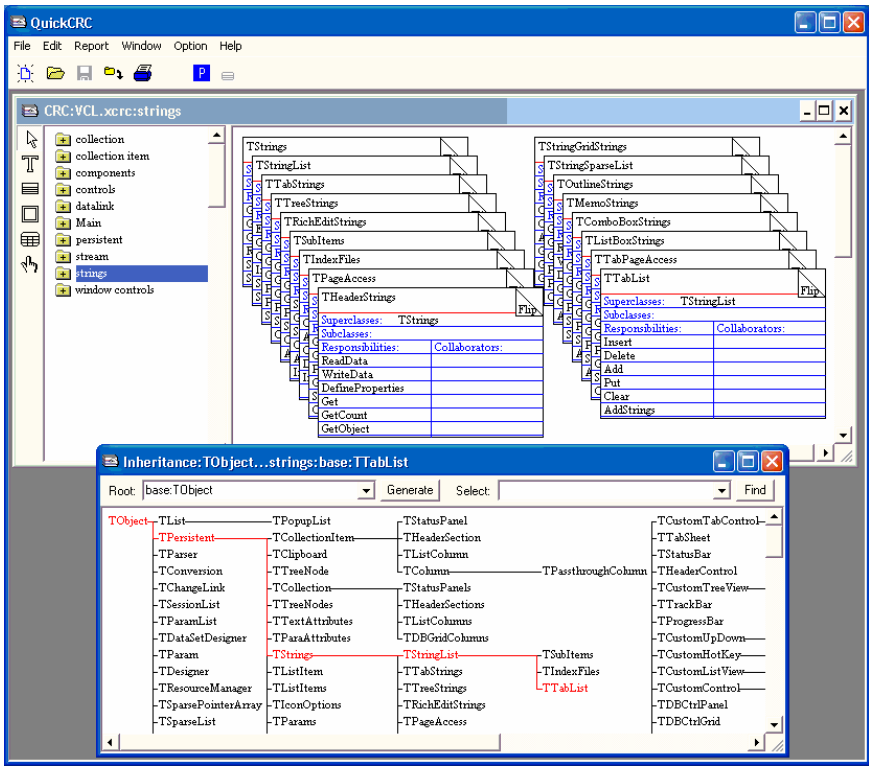
Visual Paradigm for the Unified Modeling Language (VP-UML) is a UML CASE tool. The tool is designed for a wide range of users, including Software Engineers, System Analysts, Business Analysts and System Architects interested in building reliable, large scale software systems through the use of the Object-Oriented approach. VP-UML supports the latest standards of Java and UML notations and provides the industry's full round-trip code generation and code reverse engineering support for Java.
In addition, VP-UML is integrated with Eclipse/IBM WebSphere®, Borland® JBuilder®, NetBeans IDE/Sun™ ONE, IntelliJ IDEA™, Oracle JDeveloper and BEA WebLogic Workshop™ to support the implementation phase of software development. The transitions from analysis to design and then to implementation are seamlessly integrated within the CASE tool, thus significantly reducing efforts in all stages of the software development lifecycle. [8]



Visual paradigm is a simple diagram tool. The CRC diagrams in the tool are not linked to the sequence diagrams and there isn't any other way to "do scenarios."

2.1.2. QuickCRC for Windows

QuickCRC is a software design tool ensuring the responsibility driven design of object-oriented software. CRC cards are used to discover and document classes, responsibilities, attributes and collaborations between classes. They are popular among developers using agile methods or as a front end to the UML modeling notation [7].



A screen shot of QuickCRC

We were unable to download the tool.

The responsibilities found in the scenarios are linked to the CRC cards.

Based on [6], the tool has the following features which are relevant to our paper:

- A scenario can call sub-scenarios.
- CRC cards can be grouped by function.

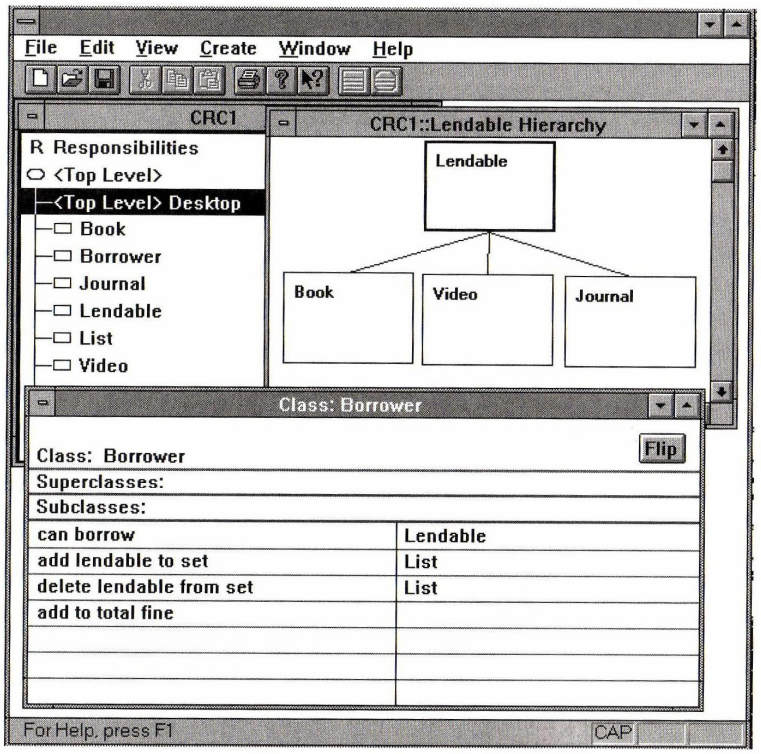
QuickCRC can also perform error checks to locate design problems. For example, responsibilities that are not used in any scenarios may indicate that the design is incomplete or perhaps the responsibility isn't needed. Likewise, a card not used by any collaboration is not needed.

2.1.3. Rose CRC

From [5]

Rose CRC

Rose CRC supports the CRC card process at various levels. It can be used simply to record classes and their responsibilities, collaborators, and superclass/subclass relationships. This is done by entering text on a window that resembles an index card. This "card" can even be flipped so that comments and descriptions can be written on the back. The tool facilitates the process of filling in the card by providing drop-down lists of previously defined classes to use for collaborators, superclasses, and subclasses. Respon-



sibilities can be detailed in a separate window where the set of operations that carries out this responsibility can be specified.

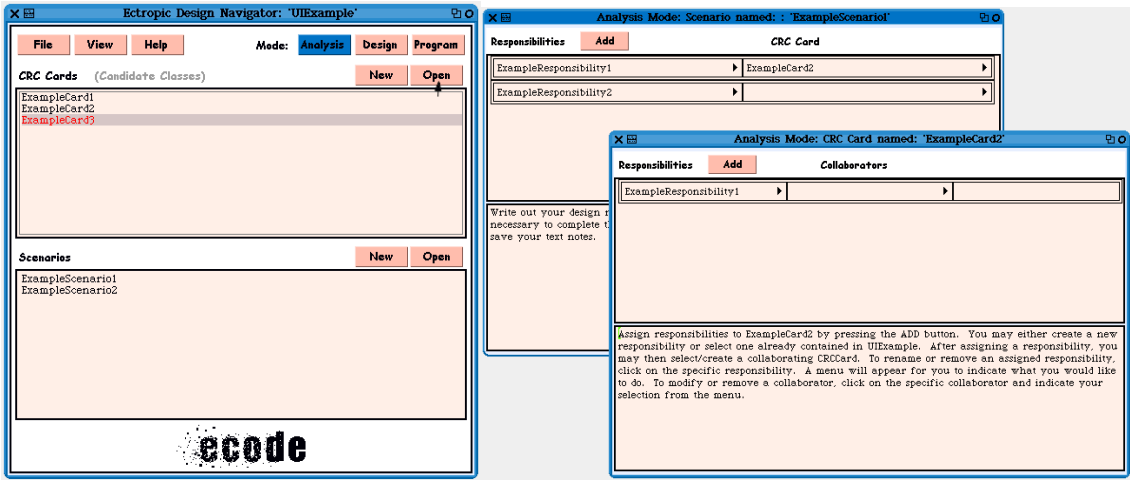
The classes can be viewed in various ways: as single cards, in a window showing all cards, or in a browser window, where the names of cards are listed, grouped by subsystem. The views are highly connected, and navigating among them is easy. New versions of the cards can be printed in various formats, including "real card" format of various sizes.

The tool also incorporates some of the notions of Wirfs-Brock's responsibility-driven design. If a user wants to group cards into named subsystems or responsibilities into contracts, the tool supports this. It will also create hierarchy graphs for any class. If the **Rose CRC** user wants to use CRC card information as input to the Booch methodology, he or she can export the model from this tool to Rational Rose. Figure 5.8 shows the part of the **Rose CRC** display for the Library application, including the Borrower CRC card and a hierarchy graph of the Lendable classes.

2.2. Free\Academic

2.2.1. ECODE

Ectropic Design is a collaborative design method, patterned on Open Source software development, by which order and structure are created from the efforts of multiple, potentially unrelated, software developers - a feature-oriented design method. Software evolves ectropically [1] through the continuous augmentation of its features, bound to specific program goals. These evolving features are defined in terms of the end-user goals the features achieve and how the features interact, both statistically and dynamically, with other features. Ectropic Design facilitates the development of software by multiple, unrelated developers working concurrently. By binding source code and collaboration technology to indented program goals, Ectropic Design provides developers with the necessary mechanisms to enhance software continuously, while maintaining the conceptual integrity of the program [10].

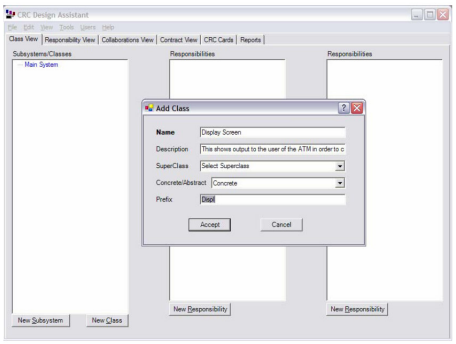


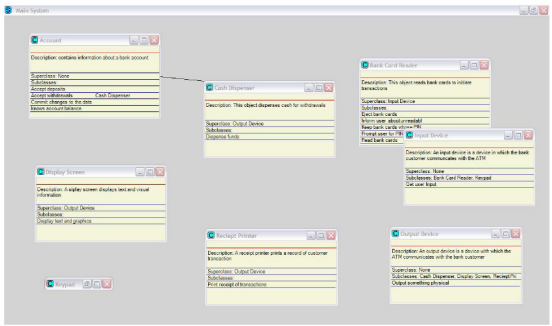
2.2.2. The CRC Design Assistant

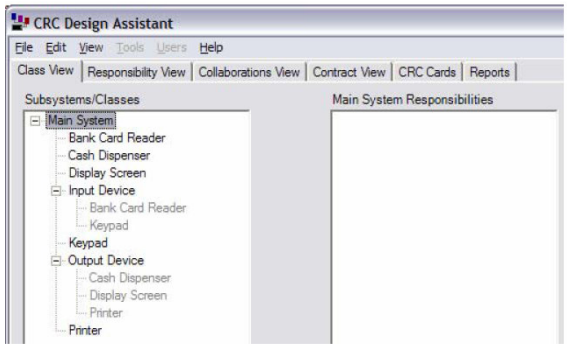
The CRC Design Assistant was first conceived to assist students in creating real-world software applications in the context of the software engineering course at UTEP. Few of the students had the experience needed to construct an efficient modular design. Specifically, students have difficulty visualizing the design, tracking design changes, maintaining the list of system responsibilities when classes are removed, revising documentation when designs change, and presenting a design to other team members and observers. The purpose of this tool is to help alleviate these problems. The key requirements of the system are:

- The system must support the CRC design method and assist students in creating, modifying, and documenting.
- Object-oriented designs.
- The system must support a team environment and be accessible over the internet.
- The system must be able to track changes made by each team member.
- The system must automatically generate design documentation in the form of RTF files and diagrams.
- The entire design must be stored in a single database.
- The system must be easy to maintain.

[9]







3. The "Easy CRC" Tool

Our study began with an actual need for a computerized tool that can be used by students during the OOSE design courses. We wanted to follow the process as described in [5] from the first stages of finding the objects from a definition test; then to the use of CRC cards to mimic sequence scenarios to the stage of class definition from the CRC cards; then to the export of the results, to be edited through more advanced tools. The CRC card use is usually in the first stages of the design process – the stage where the user identifies the base objects, responsibilities and methods of the objects. We wanted a tool that will help the student with these complex tasks, but will not be too complicated, to be used easily with little time spent on learning how to use it.

The use of the tool is splintered into two major stages – identifying the objects (CRC cards) from plain text and then identifying the collaborators and responsibilities by simulating scenarios through sequence diagrams.

3.1. Identifying Object from a Language Description

One of the issues that interested us in our efforts is a subject that was surveyed before by Abbott [11] – program design by informal plain language description.

We wanted to try to automate the process as much as possible, though it was clear the process couldn't be completely automated. Such a holistic automation would require that the computer be able to conceptually understand the requirements text, and this is a task much larger than the scope of our study.

Our idea was to identify the nouns in the text and then offer the user of the tool a convenient method for selecting which of the discovered nouns are relevant and which are irrelevant.

Associating common nouns with data types makes the notion of data types more intuitive. Most people already have an intuitive understanding of common nouns [11].

3.1.1. Deciding Which Word is a Noun

A challenge of this stage was to "sort out" the noun words from other words in the requirement text.

We found that the most effective method was to refer to a dictionary. Thus, we started by querying internet-based dictionaries. The first dictionary we used was the web site **dictionary.com**. The problem with **dictionary.com**, is that it is only in English; and we wanted to support text analysis in additional languages, such as Hebrew. Therefore, we found a web-based multi-language dictionary at the web site **babylon.com**. After trying to use it to query each word in the text, we discovered that, because of our too frequent

queries, the site seemed to crash – an unexpected and unwanted result. Moreover, the problem with web-based queries is that they take a lot of time. When the input is a rather long text, it can take such an unreasonable amount of time that the tool would be unusable.

A second approach to finding the nouns was to use a dictionary that already exists on the computer. Surprisingly, we found an existing dictionary that comes with every copy of MS Word. The Hebrew-English dictionary included with the Hebrew version of MS Word contains the classification if a word is a noun for each word in the dictionary. The dictionary is stored in plain XML format on the computer and we found it very easy to access. The dictionary seems to be exactly the same dictionary as in the website **morfix.co.il**. We didn't find any definition in the license of MS Word prohibiting us from using the dictionary in this manner, as long as a valid version of MS Word is installed on the computer. When using the MS Word dictionary, the noun classification process time was dramatically decreased to an almost instant processing, even for relatively long texts.

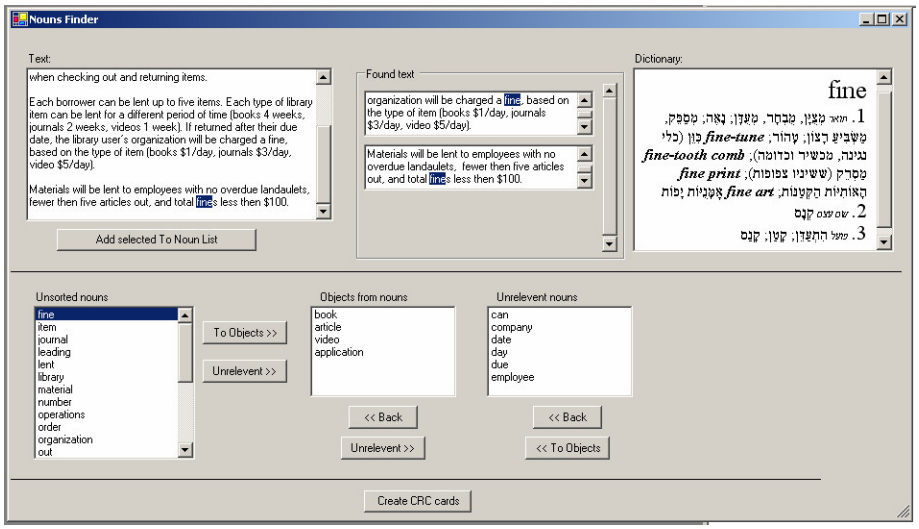
Because the MS Word dictionary only contains the words in "plain" state, we added handling of "ing" and "s" roles to find more nouns in other states.

3.1.2. Deciding Which Noun is an Object

At this stage, with the help of the tool, a long list of nouns is discovered and is displayed to the user [see figure 1]. The user needs to decide for each noun if it is a relevant object or not. To help the user better understand if the noun is an object or not, the tool will provide two methods to help.

When a noun is selected in the list, a window appears, containing the word within the text, within the line, along with preceding text and following text. This makes it easier for the user to see all the occurrences of the word [see figure 1].

The dictionary definition of the word is displayed in a small window to help with the meaning of the word.



3.2. CRC Card Diagram Editing

CRC card editing is one of the main parts of our study. The goal was to develop a graphical, easy-to-use diagram building tool enabling students to create CRC card scenarios in a manner similar to textbook CRC creation.

3.2.1. Card Editing

Editing the CRC cards using the CRC card editor can be done in different ways.
One method is to enter all information manually - editing the different values through the property editor in the tool. Right-click the "CRC cards" node in the solution explorer; a sub menu will be displayed with an option to add an item to the sub tree below. In this manner the user can Assign Responsibilities, Add Attributes, Add Methods to responsibilities and set the different properties of the item.
Another way is to create sequence diagrams and from them, the collaborators and responsibilities of the CRC cards would be discovered automatically.

3.2.2. Building Sequence Diagrams

With the tool, we first add objects in the scenario, and then we start to add the messages between the objects. For every message, there is an option to add a responsibility to the source card.

3.2.3. Exporting Data to XMI

XMI (XML Metadata Interchange) is an OMG standard for exchanging metadata information via XML. It can be used for any metadata whose metamodel can be expressed in MOF. The most common use of XMI is as an interchange format for UML models, although it can also be used for serialization of models of other languages (metamodels).

Most common UML editing software like Visual Paradigm, Argo UML, Enterprise Architect and Poseidon for UML support export and import of UML through XMI formatted files.

XMI is the common base for UML data export, so it was one of our objectives to export the results to XMI format. Different UML editing tools export data to the XMI format differently. We exported the data in the way that Visual Paradigm can import, as it is the best tool, in our opinion, for an academic course because of its features, its support for CRC cards and its free academic software licenses.

The CRC card diagram export is missing from the XMI format is. We recommend adding it to the standard.

3.3. Case Study

In this chapter we will describe the use of "Easy CRC" by means of a case study. This case study is an analysis of a simple library example. The example is taken from [4, chapter 3].

3.3.1. The Problem

This text is taken from [4, page 48]:
"The application will support the operations of a technical library for an R&D organization. This includes searching for and leading of technical library materials, including books, videos, and technical journals. Users will enter their company ids in order to use the system; and they will enter material ID numbers when checking out and returning items.

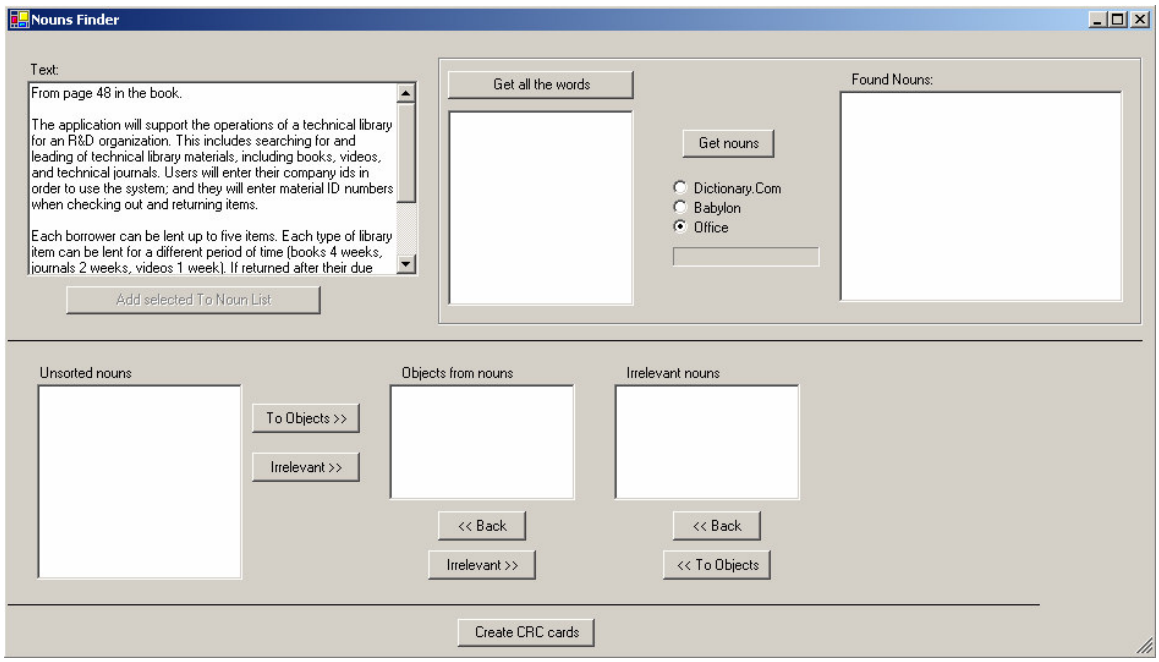
Each borrower can be lent up to five items. Each type of library item can be lent for a different period of time (books 4 weeks, journals 2 weeks, videos 1 week). If returned after their due date, the library user's organization will be charged a fine, based on the type of item (books $1/day, journals $3/day, videos $5/day).

Materials will be lent to employees with no overdue landaulets, fewer then five articles out, and total fines less then $100."

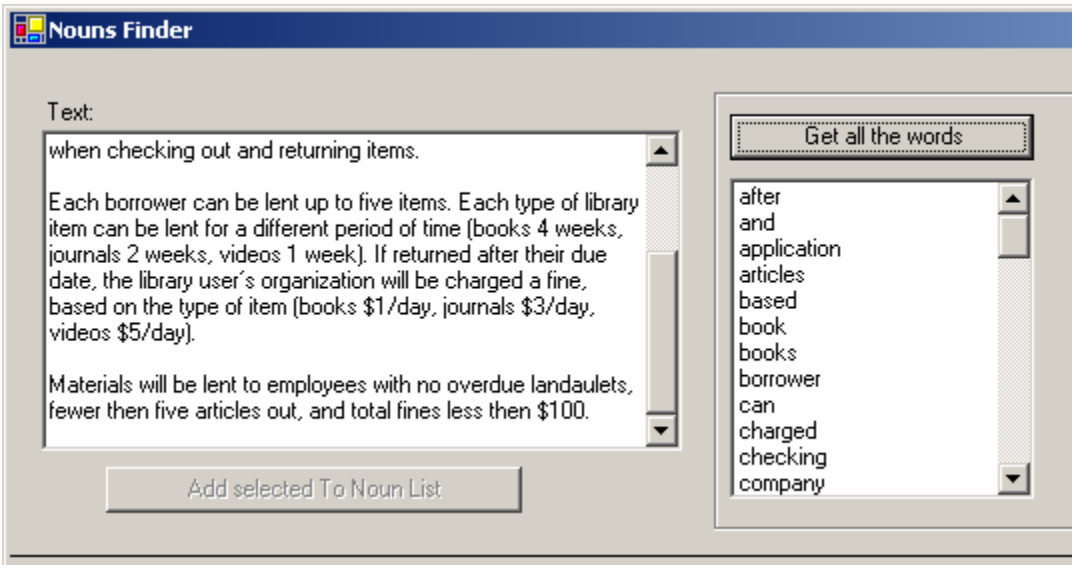
According to the book, the set of classes found in the session are: Librarian, Lendable, Book, Video, Journal, Date, Borrower, and User [4, Page 52].

3.3.2. Finding Classes From the Text

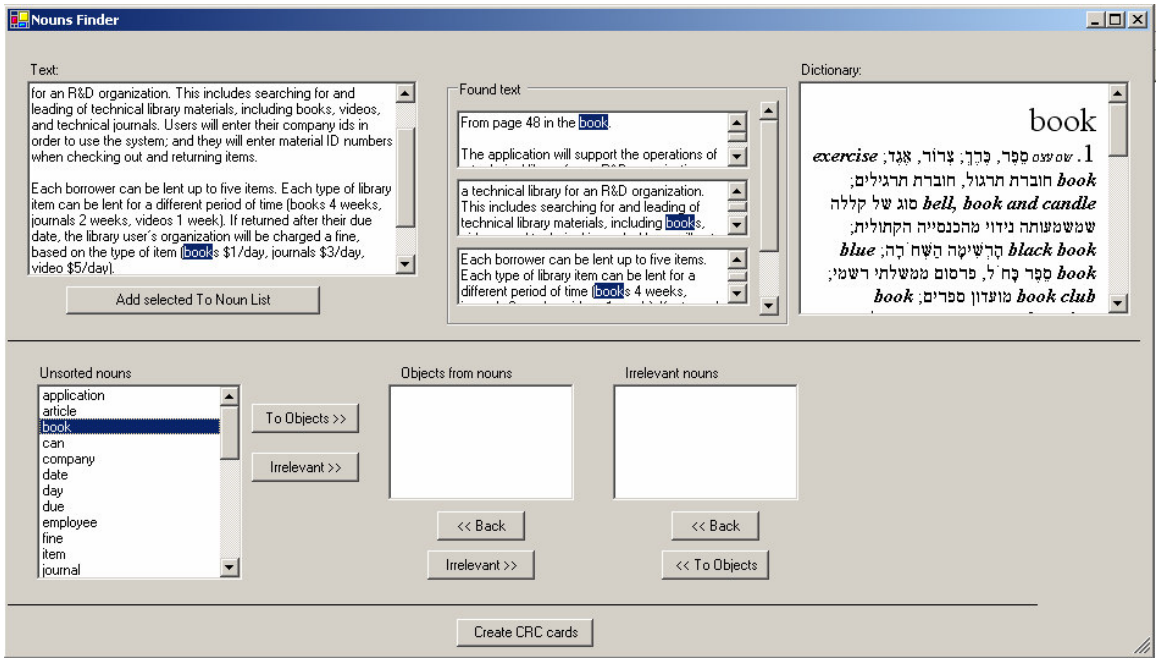
We paste or type the text into the "text" edit box in our tool.


Nouns Finder
Text:
From page 48 in the book.
The application will support the operations of a technical library for an R&D organization. This includes searching for and leading of technical library materials, including books, videos, and technical journals. Users will enter their company ids in order to use the system; and they will enter material ID numbers when checking out and returning items.
Each borrower can be lent up to five items. Each type of library item can be lent for a different period of time (books 4 weeks, journals 2 weeks, videos 1 week). If returned after their due
Add selected To Noun List
Get all the words
Get nouns
Dictionary.Com
Babylon
Office
Found Nouns:
Unsorted nouns
To Objects >>
Irrelevant >>
Objects from nouns
<< Back
Irrelevant >>
Irrelevant nouns
<< Back
<< To Objects
Create CRC cards

Then, we click the “Get all the words” button to get a list of all the unique words in the text to a list.



Then we click the “Get Nouns” button to filter only the nouns from the word list. There are three methods to get the nouns: two methods take them from web sites and one uses the MS Word dictionary (Office). We will use the Word dictionary, because it is fastest.

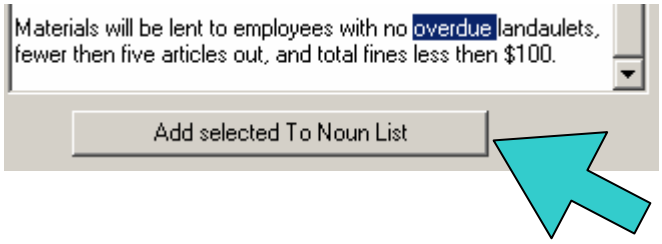


Next, we sort the nouns list for objects and irrelevant nouns.

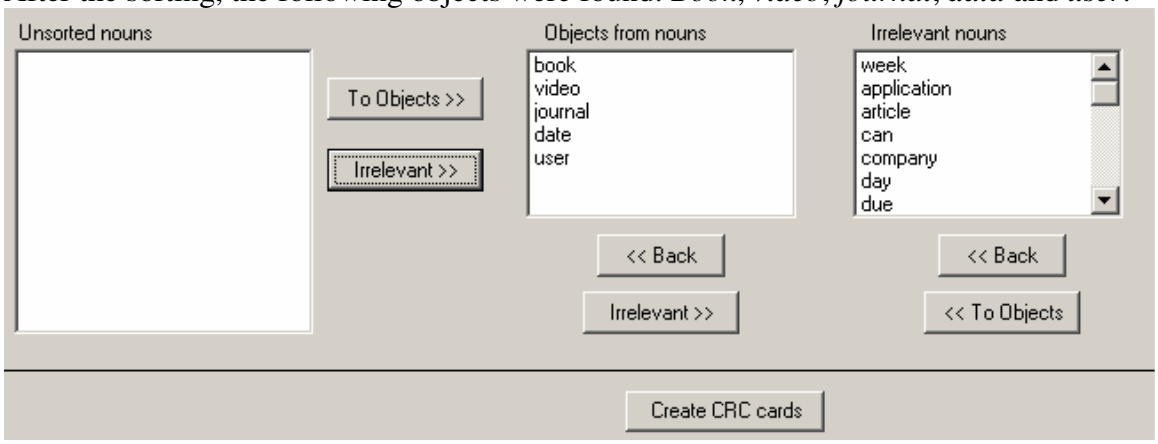
- A small dictionary window gives the dictionary translation from all the dictionaries installed with MS Office.
- A small text window will show all the references to the word in the text.

- The occurrence of the word will be highlighted in the text window.

We can also highlight objects that were not found and add them to the list.

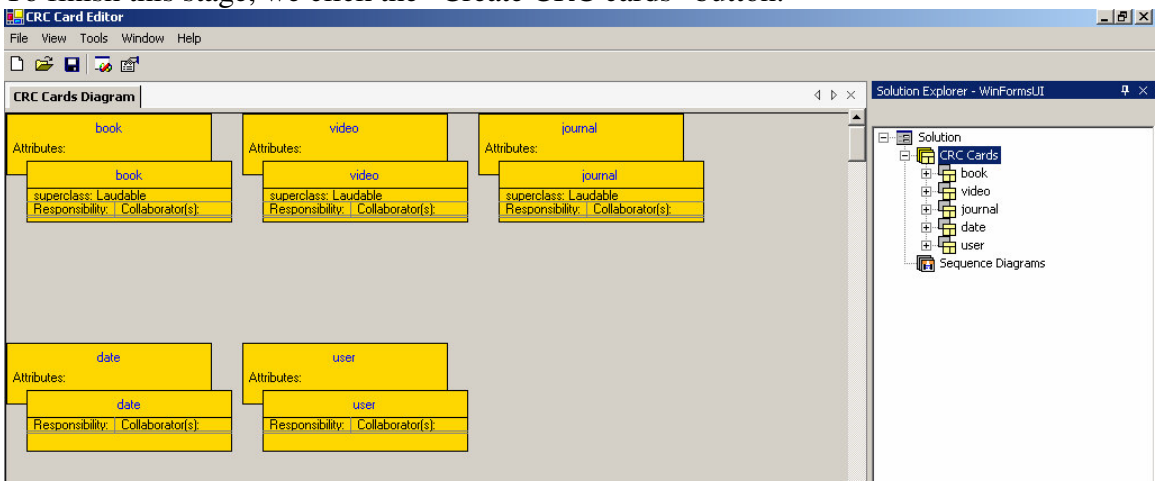


After the sorting, the following objects were found: *Book*, *video*, *journal*, *data* and *user*.



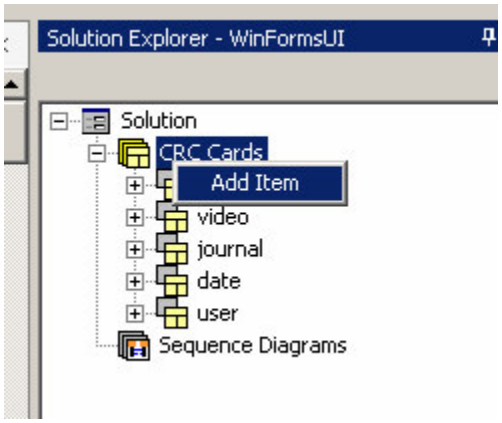
This method didn't find the following objects: *Librarian*, *Lendable*, and *Borrower* because they are not present in the text. They will be added manually later on.

To finish this stage, we click the "Create CRC cards" button.

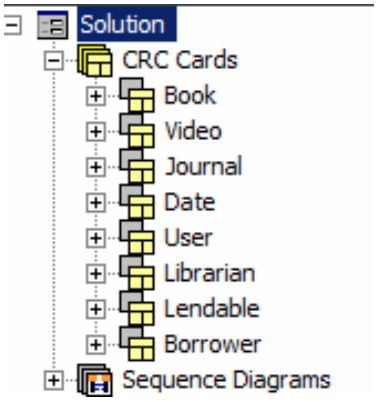


Empty CRC cards will be created in the CRC card diagram in the tool.

Next, we manually add the missing cards - Librarian, Lendable and Borrower.



The result will be like that:

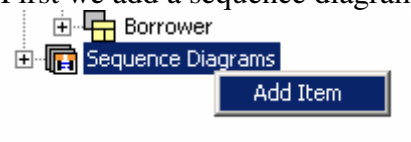


3.4. Building a Sequence Diagram

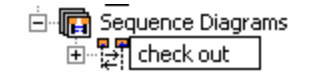
By building sequence diagrams, the user can mimic the scenario reenactment process that takes place during the brain storming.

We will recreate the sequence diagram [4, page 96].

First we add a sequence diagram to the sequence diagrams list.



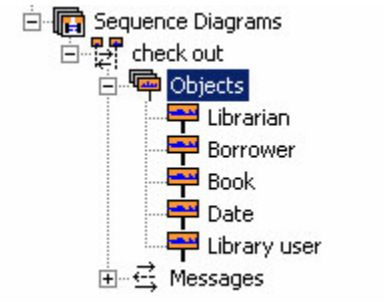
We name it “check out.”



3.4.1. Creating the Objects for the Sequence

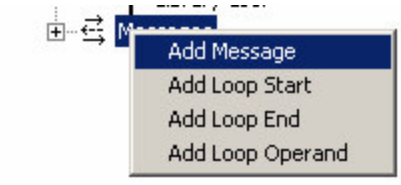
We add the five objects in the diagram, individually, to the objects node in the sequence diagram.

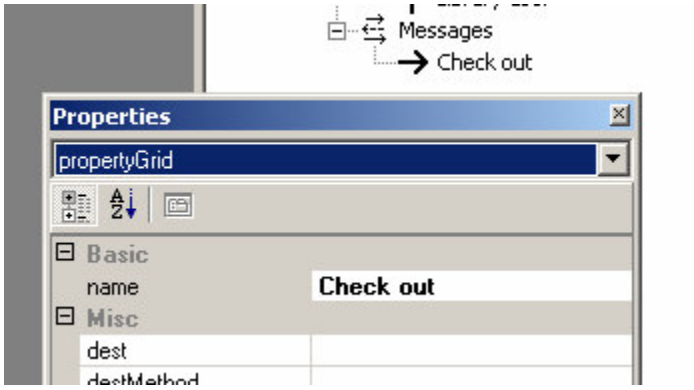
The result will be like that:



3.4.2. Adding the Messages

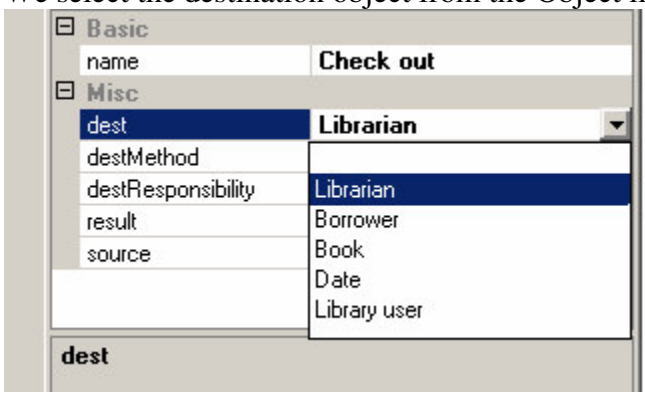
We add the messages, individually, to the Messages node.





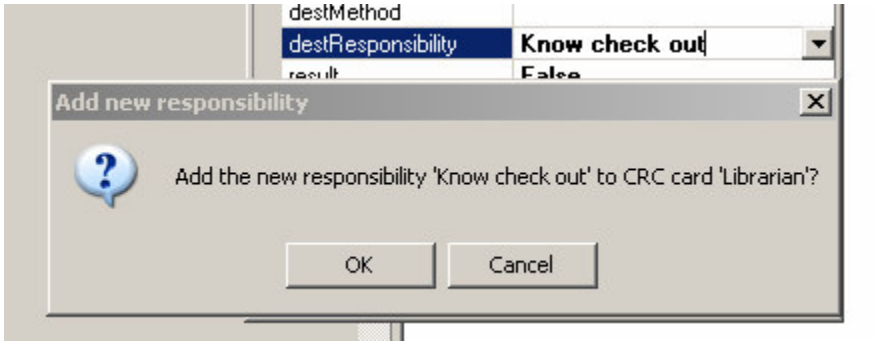
Now, we select the source object from the Object list.

We select the destination object from the Object list.



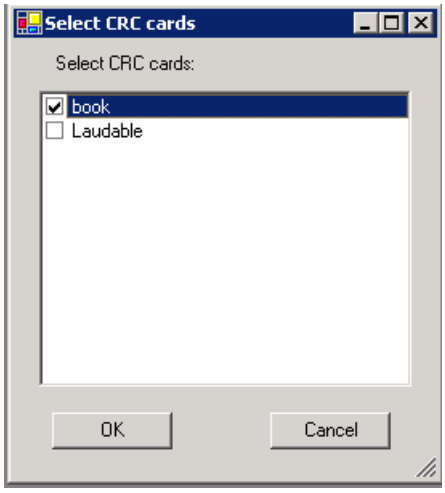
We can add the source responsibility by typing its name in the destResponsibility property.

A window opens to confirm creation of the new responsibility.



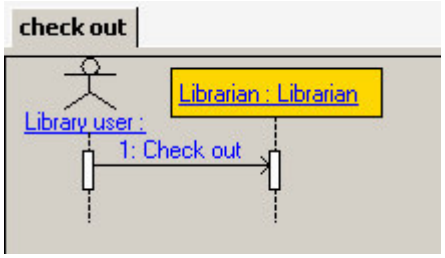
If the responsibility exists, it can be selected from a list.

If there is a superclass to the card, the hierarchy will be shown and we select which card (one or more) the responsibility will be added to.

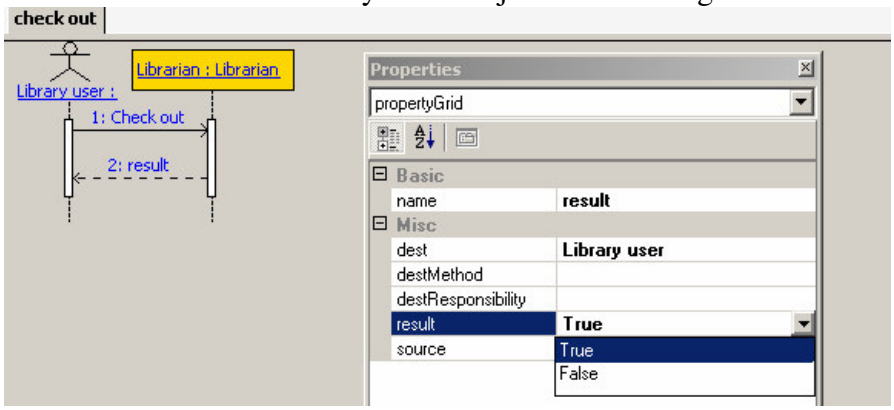


In a similar way we can add a destination method to the destination responsibility.

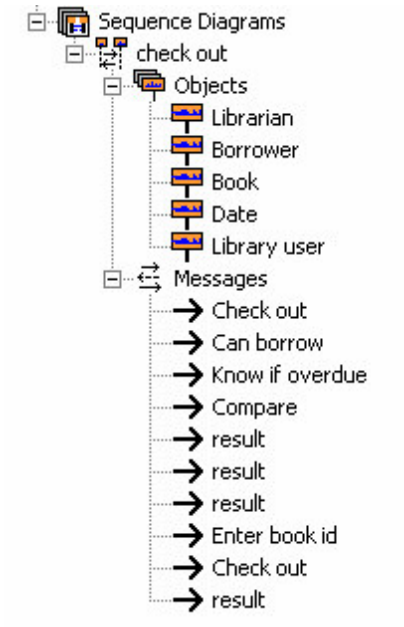
We can see the diagram being updated as we add the message.

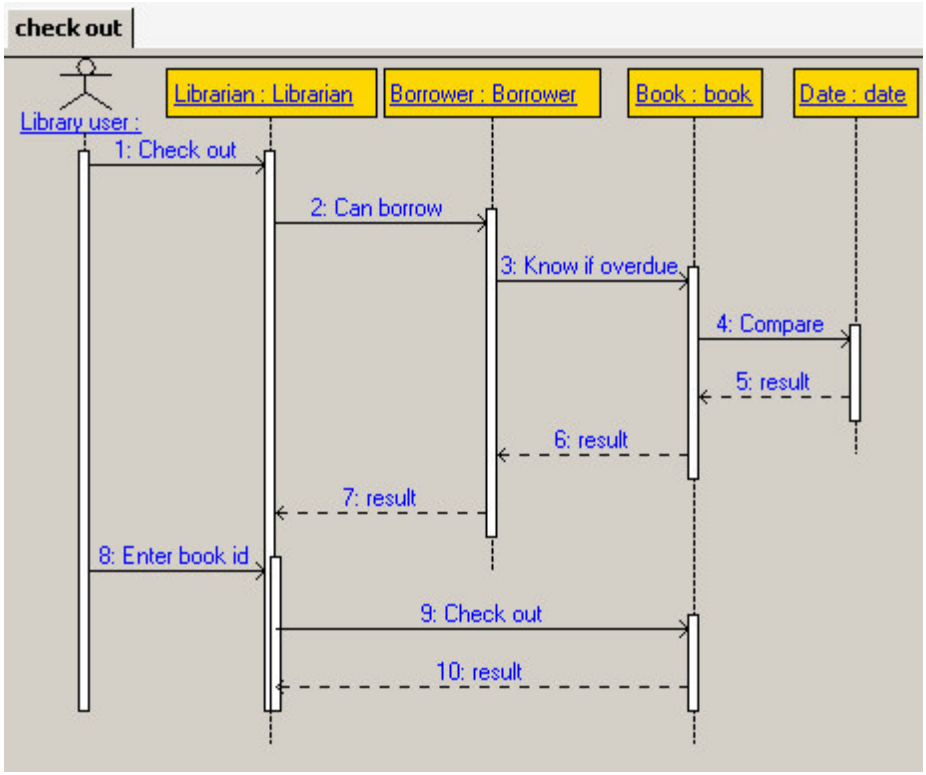


A message can be defined as a "result" by changing the "result" attribute to true – this will close the current activity of the object. The message line will be a broken line.

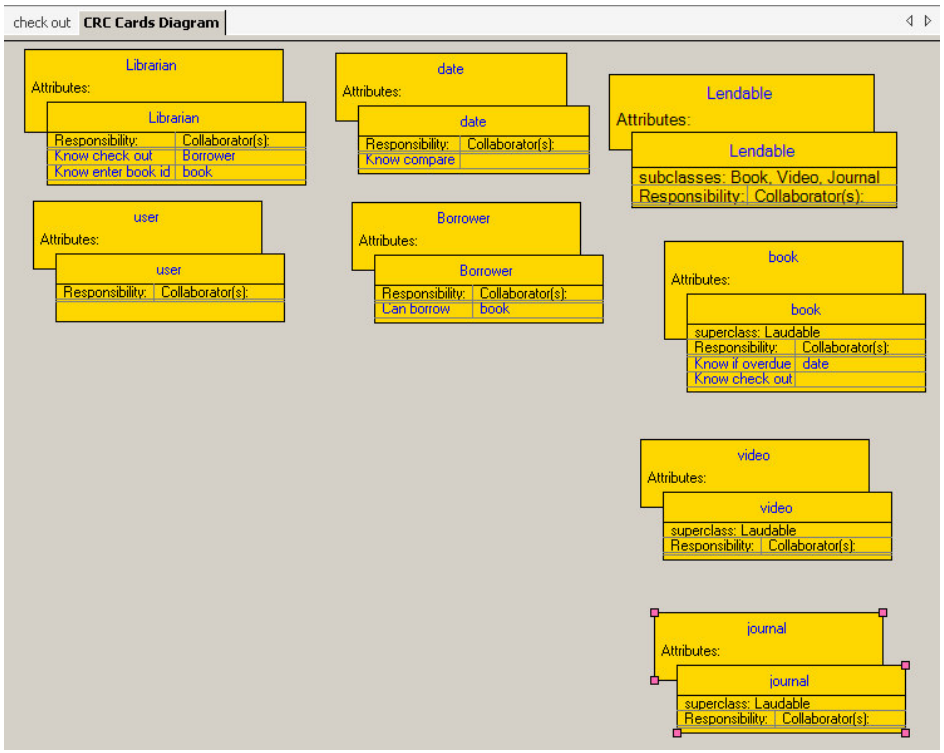


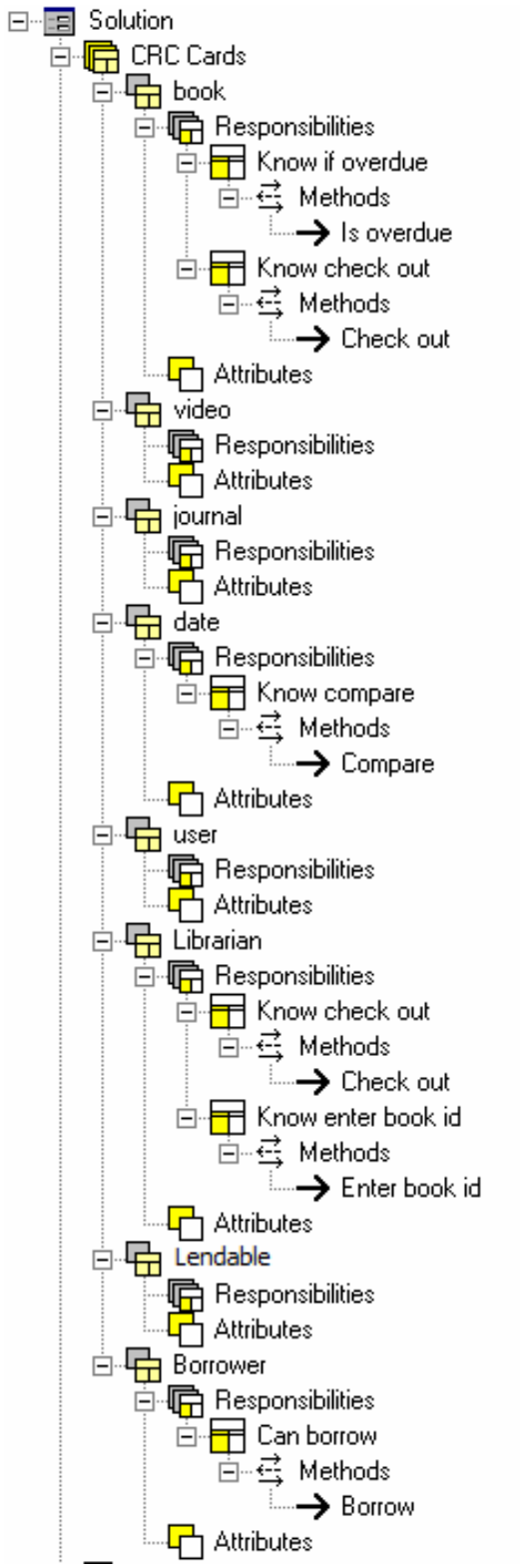
In this manner we add the rest of the messages. By doing so, we discover the responsibilities and the collaborators.





The responsibilities and collaborators were also added to the CRC cards.





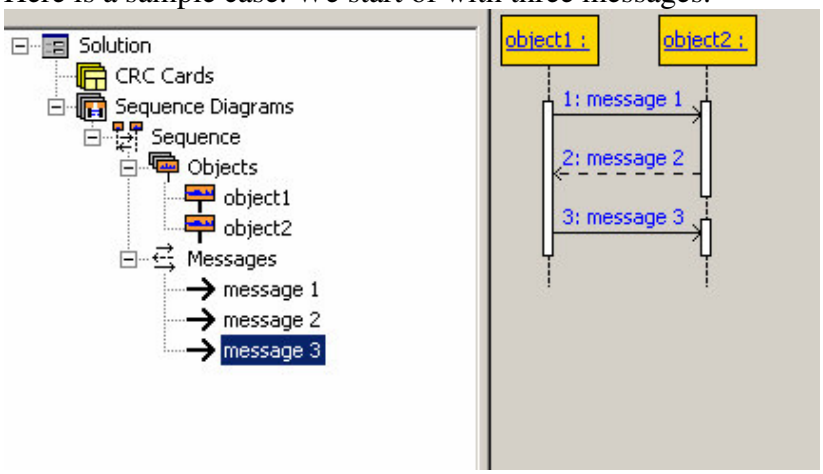
We will repeat the process for all the scenarios as sequence diagrams.

3.5. Advanced Sequence Messages

3.5.1. Loops

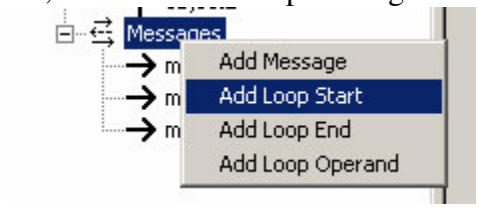
Loops can be defined in the sequence diagrams.
To create a loop in the sequence we need to add a start loop message and an end loop message.

Here is a sample case. We start of with three messages.

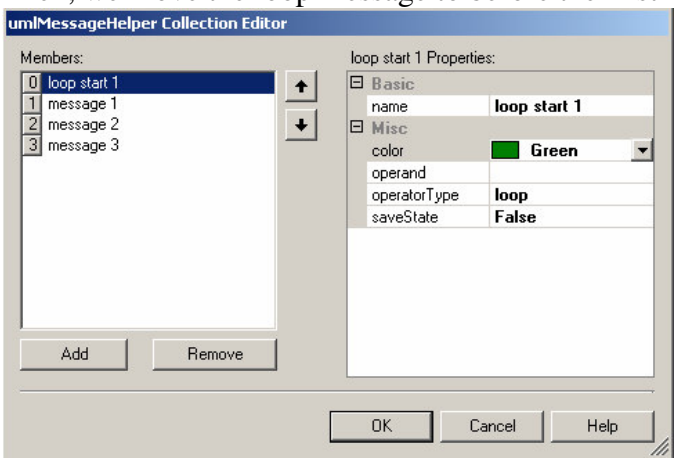


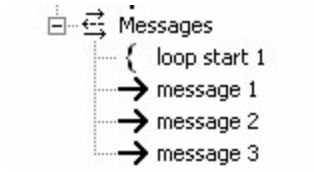
Let's say we want to loop the first two messages.

First, we add a start loop message

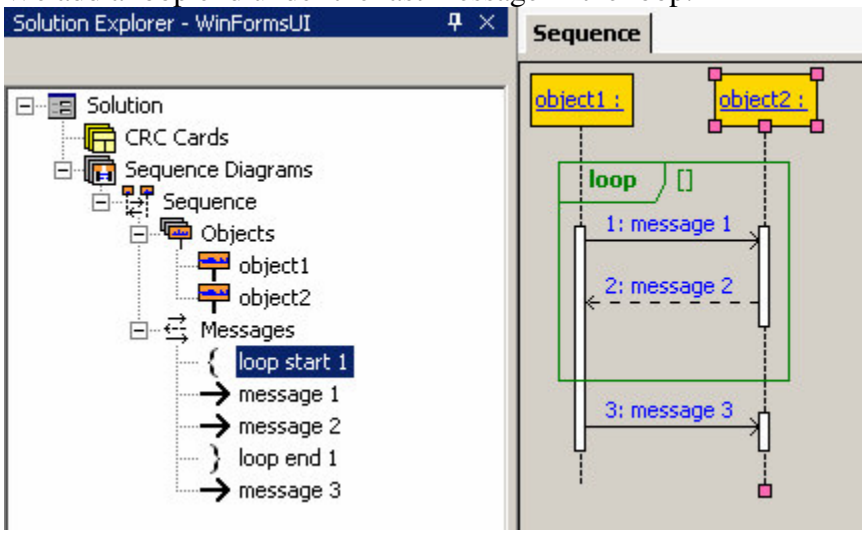


Then, we move the loop message to before the first message in the loop.

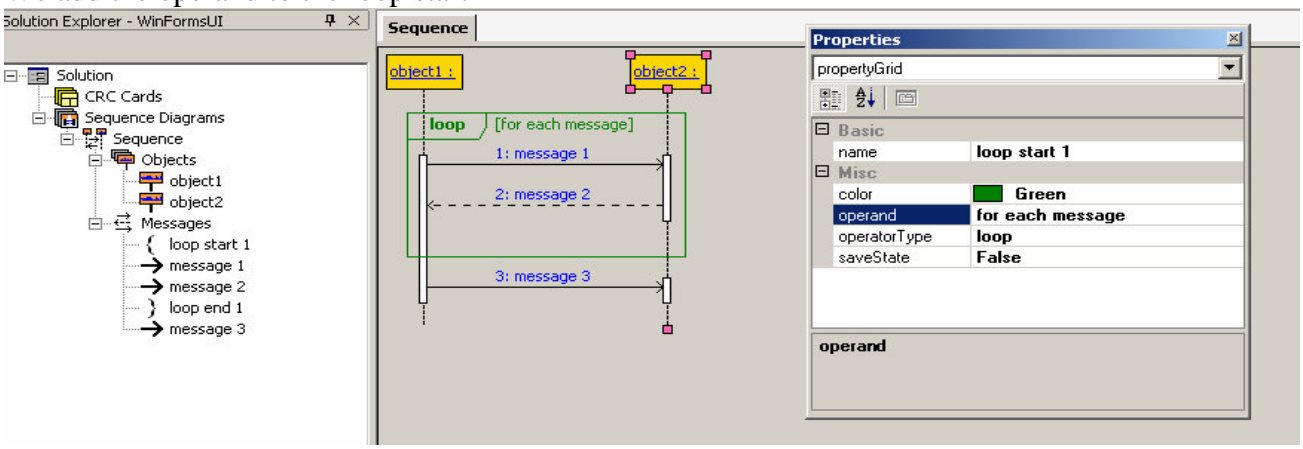




We add a loop end under the last message in the loop.



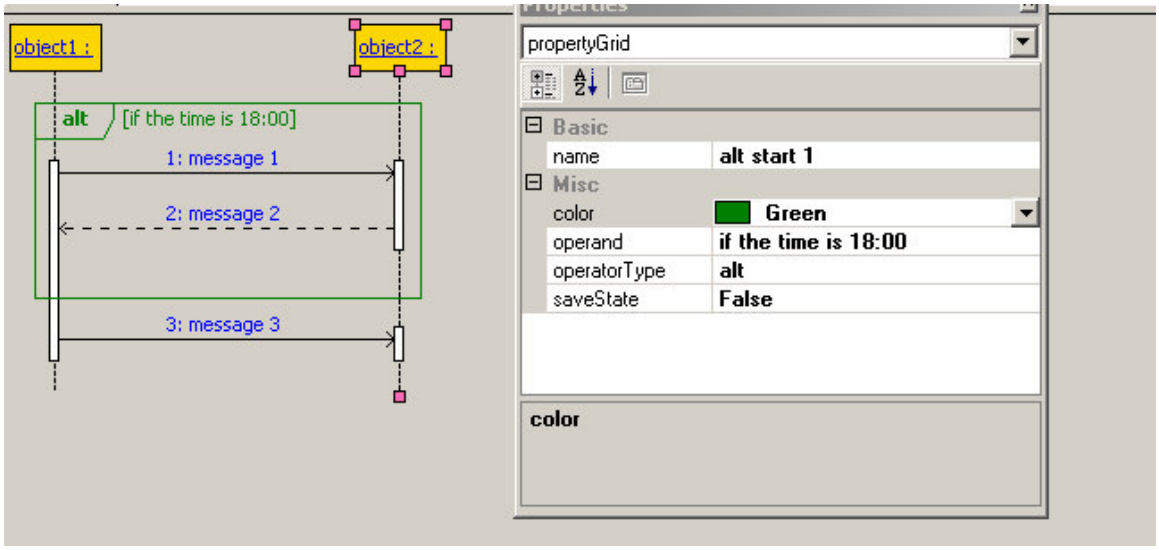
We add the operand to the loop start



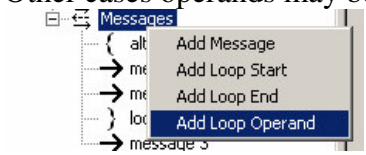
3.5.2. Case

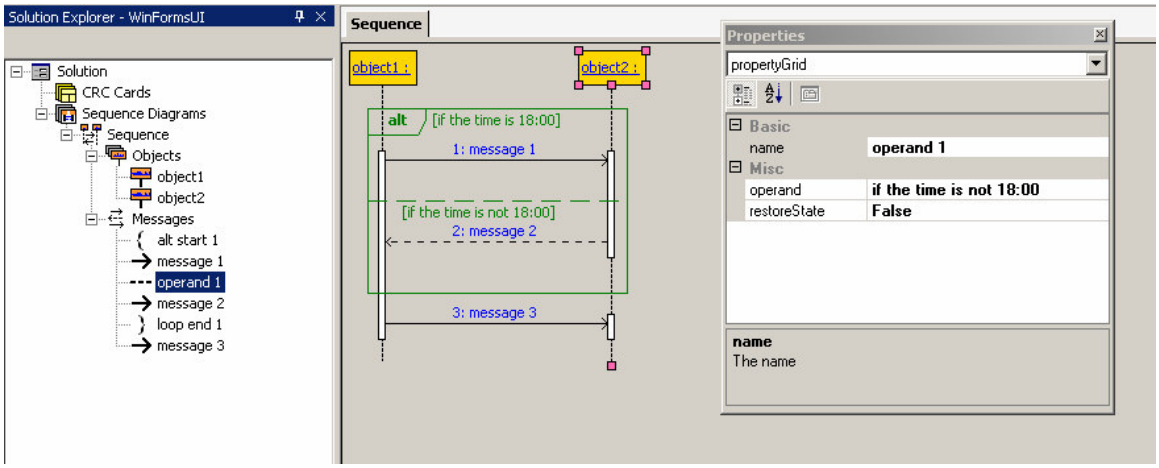
A loop can be transformed into a case, by changing the operator type of the loop start.

The operand will be the case condition.



Other cases operands may be added by adding “loop operand”.





3.6. Export to Other Tools

3.6.1. Export to Visual Paradigm

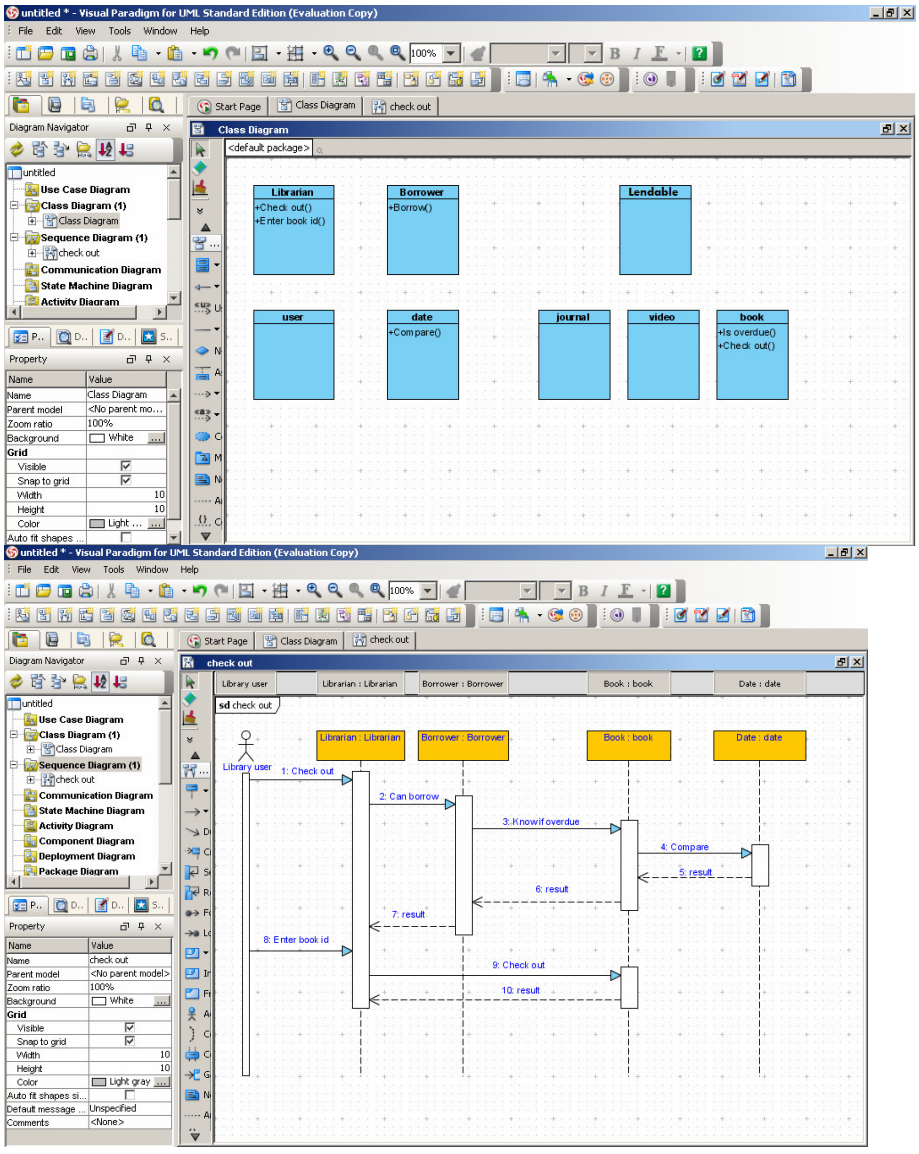
Our tool has an option to export the data to the XMI format so it can be imported to visual paradigm and other tools.

First, we select “Export to XMI” from the file menu in the tool.

Then, we open Visual Paradigm and import the XMI file (using their import command).

Because UML XMI format doesn’t support the CRC card export, we export the CRC cards as classes.

The diagram below is the Visual Paradigm output when exporting the sample above.


untitled * - Visual Paradigm for UML Standard Edition (Evaluation Copy)
File Edit View Tools Window Help
Start Page
Class Diagram
check out
Diagram Navigator
Class Diagram
<default package>
Librarian
+Check out()
+Enter book id()
Borrower
+Borrow()
Lendable
user
date
+Compare()
journal
video
book
+Is overdue()
+Check out()
Property
Name
Value
Class Diagram
<No parent mo...
100%
White
Grid
Visible
Snap to grid
Width
Height
Color
Auto fit shapes ...
untitled * - Visual Paradigm for UML Standard Edition (Evaluation Copy)
check out
Library user
Librarian : Librarian
Borrower : Borrower
Book : book
Date : date
sd check out
1: Check out
2: Can borrow
3: Know if overdue
4: Compare
5: result
6: result
7: result
8: Enter book id
9: Check out
10: result
Use Case Diagram
Class Diagram (1)
Sequence Diagram (1)
Communication Diagram
State Machine Diagram
Activity Diagram
Component Diagram
Deployment Diagram
Package Diagram
Name
Value
check out
<No parent model>
Zoom ratio
Background
White
Grid
Visible
Snap to grid
Width
Height
Color
Light gray
Auto fit shapes si...
Default message ...
Unspecified
Comments
<None>

3.6.2. Export to HTML

There is no common format that we found to support the export of a CRC card diagram. To help students edit and deliver exercises that include CRC card diagram products, we have to add an option to save the CRC card diagrams as HTML.

Here is a sample output:

book
Description:
Remark:
Attributes:

book	
superclass: Lendable	
Responsibility:	Collaborator(s):
Know if overdue	
Is overdue	
Know check out	
Check out	

This card is used in the following sequence diagrams:

- check out

Responsibilities used in sequence diagrams:

- Know if overdue
    1. check out
- Know check out
    1. check out

---

video

Description:

Remark:

Attributes:

video	
superclass: Lendable	
Responsibility:	Collaborator(s):

---

journal

Description:

Remark:

Attributes:

journal	
superclass: Lendable	
Responsibility:	Collaborator(s):

---

date
Description:
Remark:
Attributes:

date	
Responsibility:	Collaborator(s):
Know compare	
Compare	

This card is used in the following sequence diagrams:

- check out

Responsibilities used in sequence diagrams:

- Know compare
    1. check out

---

user
Description:
Remark:
Attributes:

user	
Responsibility:	Collaborator(s):

Librarian
Description: Remark: Attributes:

Librarian	
Responsibility:	Collaborator(s):
Know check out	
Check out	
Know enter book id	
Enter book id	

This card is used in the following sequence diagrams:

- check out

Responsibilities used in sequence diagrams:

- Know check out
    1. check out
- Know enter book id
    1. check out

Lendable

Description:

Remark:

Attributes:

Lendable

subclasses:

book
video
journal

Responsibility:	Collaborator(s):

Borrower
Description:
Remark:
Attributes:

Borrower	
Responsibility:	Collaborator(s):
Can borrow	
Borrow	

This card is used in the following sequence diagrams:

- check out

Responsibilities used in sequence diagrams:

- Can borrow
    1. check out

---

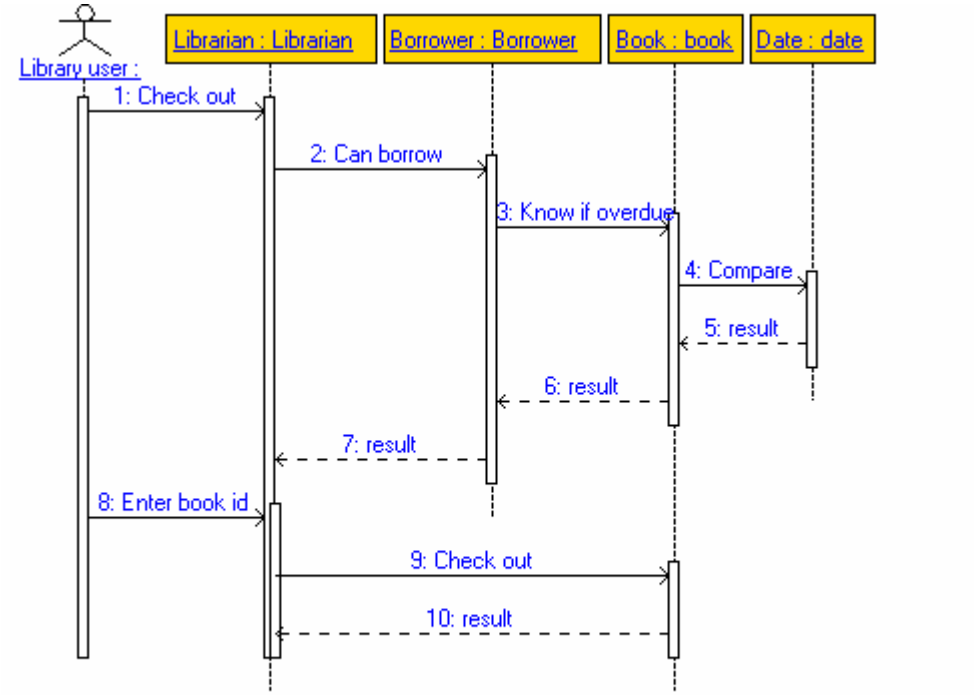
check out

Actors:

- Library user :

Objects:

- Librarian : Librarian
- Borrower : Borrower
- Book : book
- Date : date



4. Conclusion and Future Work

We envision that EasyCRC will help the students in the OOSE class to study and enable them to understand the meaning of CRC cards use faster, thus providing a better design of a system. Though the idea of CRC cards is very simple, students need extensive time for proper use of them. We believe that tools like EasyCRC will help them to adapt the CRC method.

This semester (Fall 2006) we have used the tool for the first time in three OOSE classes. Based on their experience and remarks, we intend to modify and elaborate the first version of the tool.
We would like also to enable running the scenarios as a simulation to enable an easier checking of the correctness of the scenarios and hence of the analysis and design.
The tool will be open for use, and thus we hope that (1) it will be useful for a large group of novice users, and (2) a larger community of users will help to add desired features based on their remarks.

We also recommend that the XMI UML format will be extended to support CRC cards so CRC cards analysis can be shared among tools.

5. References

1. Börstler, J. and Schulte, C. Teaching object oriented modelling with crc-cards and role playing games, Proceedings of the 8th IFIP World Conference on Computers in Education (WCCE 2005), University of Stellenbosch, Cape Town SA, July, 2005). Available at http://www.inf.fu-berlin.de/inst/ag-ddi/docs/CRCandRoleplayWCCE2005.pdf

2. Beck, K. CRC: Finding objects the easy way. Object Magazine 3(4): 42–44, 1993.

3. Object Oriented Analysis and Design using CRC Cards, http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/crc_b/ , last visited December 10, 2006.

4. Biddle, R., Noble, J. and Tempero, E. 2002. Reflections on CRC cards and OOSE design. Proceedings Tools Pacific'02: 201–205.

5. Wilkinson, N. Using CRC Cards, An Informal Approach to Object-Oriented Development. SIGS, New York, 1995.

6. QuickCRC, Introduction, http://www.excelsoftware.com/quickcrcintro.html, last visited December 10, 2006.

7. QuickCRC, http://www.excelsoftware.com/quickcrcwin.html, last visited December 10, 2006.

8. Visual Paradigm, http://www.visual-paradigm.com/, last visited December 10,2006.

9. Roach, S. and J. Vasquez, J., A Tool to Support CRC Design, Proceedings of the International Conference on Engineering Education, Gainesville, Florida, October, 2004. CRC Design Assistant Tool available at http://www.cs.utep.edu/sroach/crcda.html, last visited December 10, 2006.

10. Ectropic Software, http://www-static.cc.gatech.edu/computing/ectropic/, Aug 18 2006, http://72.14.221.104/search?q=cache:MgqzvBY0XL0J:www-static.cc.gatech.edu/computing/ectropic/+%22Imagine+hundreds+of+people+working+on+the+same+program+at+the+same+time.+Imagine+%22&hl=en&ct=clnk&cd=1

11. Abbott, R. J., Program design by informal English descriptions, Communications of the ACM, 26 (11): 882 – 894, November 1983.

12. Objects by Design, http://www.objectsbydesign.com/books/RebeccaWirfs-Brock.html, last visited December 10, 2006.