Abstract
CRC cards are an informal approach to object oriented modeling, used to represent the responsibilities of classes and the interaction between the classes. They are created through white-box scenarios that model the behavior of the system. Sequence diagrams are the most popular UML artifact for scenarios modeling. Though closely related, the CRC cards methodology is not supported by most UML modeling tools, and tools that support CRC cards do not support UML diagrams.

This paper presents EasyCRC, a tool that combines sequence diagrams with CRC cards, originally written to teach the object-oriented paradigm.

1. Introduction
The CRC cards technique has been offered by Cunningham and Beck [2]. Those are index cards which characterize each class by its name, the responsibilities of the class, and the names of other classes with which the class collaborates. CRC cards provide a simple approach to identify classes for an object-oriented development project. They are very helpful in the analysis and design phases of the development, especially for people with no experience in object-oriented analysis and design. As such, it is a suitable approach for students that participate object-oriented software engineering (OOSE) classes. And indeed, CRC cards have been adopted by many educators to teach early design in their object-oriented programming courses (e.g., [3,4,8]).

We have identified the need for a good tool to help students in the OOSE course. In this course, students are exposed to a plethora of new ideas, and frequently the students encounter difficulties in preparing assignments. Using the CRC approach, the students are asked to identify the candidate classes (the Abbott textual analysis technique [1] is a good way to identify those classes from various texts such as the problem description, use cases, etc.); to describe white-box scenarios (based on use cases) in order to find the responsibilities of the classes; and to identify the collaborations needed to fulfill the responsibilities.

1.1. Motivation
Several tools enable the creation of CRC cards (e.g., [11, 6, 5]), There exist a lot of tools that assist in creating
sequence diagrams. So why bother and create yet another tool? After reviewing tools that follow the CRC cards approach (see Section 1.3), several reasons convinced us that a new tool, and a new approach, is needed.

CRC cards are not an integral part of UML. Therefore, most tools do not provide UML diagrams (for our purpose, the sequence diagrams are of major importance) together with CRC cards\(^1\). Quoting Rebecca Wirfs-Brock [9] “… Why haven't UML tools provided any support for CRC cards? I would prefer to rephrase this as why haven't UML tools provided good support for expressing the behavior and characteristics of model elements at different levels of abstraction. I think this is because most tools (especially those supporting roundtrip engineering) are focused on providing an implementation level view of a class. Fowler talked about three levels (sure there could be more, but this is good enough to get the idea) of abstraction: a conceptual view, a specification view and an implementation view of your design. Well CRC card-level descriptions are especially good for a conceptual view. And I think most tools which are focused on turning your specification into an implementation don't see the value (or perhaps don't have a clean way to support) overlaying these different views onto a design. I wish they did and I'd be happy to work with tool vendors to get them to support different views."

Most UML tools are professional tools, which have mainly been built for the software industry. As such, while providing many useful features, it is overwhelming for the needs of an object-oriented analysis and design course. We do not need diagrams that deal with deployment, as well as translating diagrams into code. Sometimes the learning curve of such tools is too high, especially for a one semester course.

UML tools that do support the CRC cards diagram, do not link this diagram with the other ones (i.e., with the sequence diagrams). Hence, when using them, the process of creating CRC cards is mainly drawing them, and it does not follow the process as described in [13], for example. This process includes the following activities: marking nouns, removing redundant nouns, running scenarios to find responsibilities and collaborations of classes as well as missing classes.

There is no integral link between the CRC cards and the sequence diagrams. Therefore, most often changes in sequence diagram that reflect also changes in the CRC class (e.g., new responsibilities, new classes, etc.) are not followed by the appropriate changes in the CRC cards, and vice versa.

To conclude, to the best of our knowledge none of the existing tools enables the users to follow the CRC cards technique all the way from a natural language description to a complete set of CRC cards capable of being transformed into the appropriate classes. Hence, there is a need for a tool that will assist novice users, such as OOSE students. This tool should be used in the early stages of the design process. It must be easy to use and intuitive, to ensure that students' time is optimized and dedicated to learn OOSE design rather than learning how to use the tool. In addition, since some students use UML tools (which do not support the CRC cards approach), the tool should enable interoperability with existing tools, which basically means that the tool should be able to export its product to an XML format.

1.2. CRC cards

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

“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 that they provide an informal and non-threatening environment that is productive to working and learning.” [8].

A CRC card corresponds to a class. A class 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 responsibility is something the objects of a class take care of; a service provided for other objects. A responsibility can be either something the class knows or something the class does. To do something, an object may use its (local) knowledge. If this is not sufficient for the purpose, the object can collaborate with other objects: the collaborators. Responsibilities are written along the left side of the card; collaborators are written along the right side, in the same row as the corresponding responsibility.

The back of the card can be used for the class’ description and attributes.

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; the labels on the interaction arrows are the responsibilities assigned to that object to carry out various sub-tasks [13, page 97].

---

\(^1\) Visual Paradigm [8] is the only tool that we know which enables to draw both CRC cards and Sequence Diagrams. However, it does not associate both drawings.
1.3. Related work

We have reviewed tools as were available through the Internet and other resources. The tools can be divided into two major groups – commercial tools and free/academic tools. Not all the tools were tested, as some of them were not available: The software versions we have used are either free or trial versions; this limited our observations. Following is a list of tools; due to space limitation, we do not supply a survey. (The full details can be found in [10].) Commercial tools: Visual Paradigm [12] and QuickCRC [6]. Free tools: ECODE [5], The CRC Design Assistant [11].

2. The EasyCRC Tool

A need arose for a tool that can be used by students during OOSE courses. We wanted to follow the complete process described in [13]: finding classes from a definition text; using CRC cards to mimic sequence scenarios; defining classes from the CRC cards. CRC cards are usually used at the first stages of the design process, where the user identifies (conceptual) classes, responsibilities, and methods. We aimed at a tool that will help the students with these tasks, yet will not be too complicated, i.e., a tool that can be used easily with little time spent on learning how to use it.

The use of the tool is divided into two major stages: a) identifying the classes (CRC cards) from plain text, and b) identifying the collaborators and responsibilities by simulating scenarios through sequence diagrams.

In the rest of this section, we describe how EasyCRC is used to analyze and design a system based on CRC cards.

2.1. Identifying classes

Abbott [1] offered a program-design based on informal plain language description. Associating common nouns with data types makes the notion of data types more intuitive. We wanted to automate the process as much as possible, though it was clear the process could not be completely automated. Such a holistic automation requires the program to be able to conceptually understand the requirements text, and this is a task outside the scope of this study.

We identify the nouns in the text and then offer the user a convenient method for selecting which of the discovered nouns are relevant and which are not.

2.1.1. Finding nouns

A challenge of this stage was to automate the sorting out of the nouns in the requirements documents from other words in text.

We found that the most effective method is to refer to a dictionary. We started by querying internet-based dictionaries. The first dictionary we used was the website http://dictionary.com. A problem we encountered was the fact that the dictionary was only in English, whereas we wanted to support text analysis in additional languages, such as Hebrew. We found a web-based multi-language dictionary at the website http://www.babylon.com. After trying to use it to query each word in the text, we discovered that, due to our too frequent queries, the site seemed to crash – an unexpected and unwanted result. Moreover, the problem with web-based queries is that they are time consuming. When the input is a rather long text, it can take an unreasonable amount of time, making the tool unusable.

Another approach for identifying nouns is to use a dictionary that already exists on the computer. We found an existing dictionary that comes with every copy of MS Office. The Hebrew-English dictionary included with the Hebrew version of MS Office contains, for each word in the dictionary, a classification whether a word is a noun. The dictionary is stored in plain XML format and we found it very easy to access. When using the MS Office dictionary, the noun classification process time was considerably decreased to an almost instant processing, even for relatively long texts. We have also integrated the use of WordNet®, a large lexical database of English that can be employed with our tool as a local nouns database (http://wordnet.princeton.edu/). However, WordNet too supports only English.

2.1.2. Deciding which noun is a class

At this stage, with the help of the tool, a long list of discovered nouns is displayed to the user. The user decides for each noun whether it is a relevant class or not. EasyCRC helps the user to better understand whether the noun is a class or not:

1) When a noun in the list is selected, a window appears, containing the word within the text, within the line, along with preceding text and following text. This enables the user to easily view all the occurrences of the word.

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

2.2. CRC card diagram editing

CRC card editing is a major part of our study. Our 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.
2.2.1. 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 and updated.

2.2.2. Building sequence diagrams

With EasyCRC, we first add objects needed for a scenario, and then we start to add the messages between the objects. For each message, there is an option to add a responsibility to the source card.

2.2.3. 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 (MetaObject Facility). 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 UML editing software such as 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, hence one of our objectives was to export the results to XMI format. Different UML editing tools export data to the XMI format differently. We found Visual Paradigm to be the most suitable tool for an academic course. This is due to its features, its support for CRC cards and its free academic software licenses. Therefore we exported the data in a format that Visual Paradigm can import.

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

3. An Example

In this section we describe the use of EasyCRC by means of a case study. This case study is an analysis of a simple library example, taken from [13, Chapter 3].

3.1. The problem

We have used a problem described in [13, page 48], which describes system that supports a technical library for a research and development organization. Some details have been omitted due to lack of space.

3.2. Finding classes from the text

We type, or paste, the text into the “text” edit box. Then, we click the “Get all the words” button to get a list of all the unique words in the text to a list. Clicking the “Get Nouns” button filters the nouns from the words list. There are three methods to get the nouns: two methods take them from web sites and one uses the MS Office dictionary. We will use the Office dictionary, as it is the fastest in finding the nouns (the dictionary is stored on the local computer).

3.1. The problem

We have used a problem described in [13, page 48], which describes system that supports a technical library for a research and development organization. Some details have been omitted due to lack of space.

3.2. Finding classes from the text

We type, or paste, the text into the “text” edit box. Then, we click the “Get all the words” button to get a list of all the unique words in the text to a list. Clicking the “Get Nouns” button filters the nouns from the words list. There are three methods to get the nouns: two methods take them from web sites and one uses the MS Office dictionary. We will use the Office dictionary, as it is the fastest in finding the nouns (the dictionary is stored on the local computer).

3.1. The problem

We have used a problem described in [13, page 48], which describes system that supports a technical library for a research and development organization. Some details have been omitted due to lack of space.

3.2. Finding classes from the text

We type, or paste, the text into the “text” edit box. Then, we click the “Get all the words” button to get a list of all the unique words in the text to a list. Clicking the “Get Nouns” button filters the nouns from the words list. There are three methods to get the nouns: two methods take them from web sites and one uses the MS Office dictionary. We will use the Office dictionary, as it is the fastest in finding the nouns (the dictionary is stored on the local computer).

Next, we sort the nouns list to select objects and remove irrelevant nouns. EasyCRC provides the following assistance: the dictionary translation for the nouns, taken from all the dictionaries installed with MS Office; all the references to the word in the text are given; the occurrence of the word is highlighted in the text window.

The user can also highlight classes that were not found and add them to the list.

After sorting, the following classes were found: Book, Video, Journal, Date, and User.
Following the example in [13], Librarian, Lendable, and Borrower were not found, since they are not included in the text. They can be added manually.

To finish this stage, we click the “Create CRC cards” button. Empty CRC cards are created in the CRC card diagram in the tool.

![Figure 4]

Note that the noun-finding step is optional, and may be used when there are written documents (requirements, use cases, etc.). CRC cards can also be added directly, e.g. when cards are discovered during the role-playing phase.

### 3.3. Building a sequence diagram

By building sequence diagrams, the user can mimic the scenario reenactment process that takes place during the brainstorming stage. First we add a sequence diagram to the sequence diagrams list, by clicking “Add Item”.

We name the sequence “check out.”

![Figure 5]

#### 3.3.1. Creating the objects for the sequence

We add the five objects found to the objects node of the sequence diagram.

![Figure 6]

#### 3.3.2. Adding the messages

We add the messages to the messages node. As can be seen in the figure, we can use loops.

![Figure 7]

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

![Figure 8]

In the same way we select the destination object from the Object list.

A responsibility can be added by typing the responsibility name in the destResponsibility property. If the responsibility already exists, it can be selected from a list.

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

![Figure 9]

A message can be defined as a “result” by setting the “result” attribute to true; this will close the current activity of the object. The message line will be a dashed 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 are added also to the CRC cards.

![Figure 10]

We repeat this process for all scenarios.

### 3.4. Export to other tools

EasyCRC can export the data to the XMI format, which can be imported to other tools (we tried it with Visual paradigm). First, “Export to XMI” is selected from the file menu. Then, we open Visual Paradigm and import the XMI file (using the import command of Visual Paradigm). Because UML XMI format does not 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.

![Figure 11]
3.5. Export to HTML

We do not know a common format that supports the export of a CRC card diagram. To help students edit and deliver exercises that include CRC card diagram products, we have added an option to save the CRC card diagrams as HTML.

Here is a sample output:

<table>
<thead>
<tr>
<th>superclass:</th>
<th>Lendable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Responsibility:</td>
<td>Collaborator(s):</td>
</tr>
<tr>
<td>Know if overdue</td>
<td>Is overdue</td>
</tr>
<tr>
<td>Know check out</td>
<td>Check out</td>
</tr>
</tbody>
</table>

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

check out sequence

Actors:

- Library user:
- librarian: Librarian
- borrower: Borrower
- book: Book
- date: Date

4. Summary 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 to learn how to use them properly. We believe that tools like EasyCRC will help them to adopt the CRC method.

In the Fall 2007 semester we have used the tool for the first time in three OOSE classes. Based on the students experience and remarks, we intend to modify and elaborate the first version of the tool.

We also intend to enable running the scenarios as a simulation, to ease the 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 hope that the XMI UML format will be extended to support CRC cards so CRC cards analysis can be shared among tools.

References

* All the URLs have last been visited on May 2007.


