CRC Cards

- A tool and method for systems analysis and design
- Part of the OO development paradigm
- Highly interactive and human-intensive
- Results in the definition of objects and classes

Adapted from (Zenebe & Miao, 2001)
HISTORY

• Introduced at OOPSLA in 1989 by Kent Beck and Ward Cunningham as an approach for teaching object-oriented design.

• In 1995, CRC cards are used extensively in teaching and exploring early design ideas.

• CRC cards have become increasingly popular in recent years. As formal methods proliferate, CRC cards have become, for some projects, the simple low-risk alternative for doing object-oriented development.

Adapted from (Zenebe & Miao, 2001)
What’s a CRC Card?

CRC stands for *Class, Responsibility*, and *Collaboration*.

- **Class**
  - A set of objects that share common structure and common behavior
    - **Super-class**: a class from which another class inherits
    - **Subclass**: a class that inherits from one or more classes

- **Responsibility**
  - Some behavior for which an object is held accountable.

- **Collaboration**
  - process whereby several objects cooperate to provide some higher-level behavior.

Adapted from (Zenebe & Miao, 2001)
What’s a CRC CARD? (Cont.)

• An index card that is annotated in a group setting to represent a class of objects, its behavior, and its interactions.
• An informal approach to OO modeling
• Created through scenarios, based on the system requirements, that model the behavior of the system.

Adapted from (Zenebe & Miao, 2001)
Sample CRC Card (Front & Back)

Adapted from (Zenebe & Miao, 2001)
REQUIREMENTS

• Cards should be physical cards, not virtual cards.
• CASE tools for support of CRC cards are useful, but cannot replace the interaction that physical cards facilitate.
• 3x5 or 4x6 inch are the perfect size. But you can really use anything you want. ….Napkins??? Envelopes???
• Refreshments (Optional)

Adapted from (Zenebe & Miao, 2001)
THE CRC CARD SESSION

• The session includes a physical simulation of the system and execution of scenarios.

• Principles of successful session
  – All ideas are potential good ideas
  – Flexibility
  – Group Dynamic
BEFORE THE SESSION

• Forming the Group
  – The ideal size for the CRC card team:
    • 5 or 6 people
  – The team should be composed of
    • One or two domain experts
    • two analysts
    • an experienced OO designer
    • one group’s leader/facilitator

Adapted from (Zenebe & Miao, 2001)
The CRC Card Team

Adapted from (Zenebe & Miao, 2001)
Source: The CRC Card Book by Bellin et.al (1997)
DURING THE SESSION

• All the group members are responsible for holding, moving and annotating one or more cards as messages fly around the system.

• Group members create, supplement, stack, and wave cards during the walk-through of scenarios.

• A session scribe writes the scenarios.

Adapted from (Zenebe & Miao, 2001)
PROCESS

1. Brainstorming
   - One useful tool is to find all of the nouns and verbs in the problem statement.

2. Class Identification
   - The list of classes will grow and then shrink as the group filters out the good ones.

3. Scenario execution (Role play)
   - The heart of the CRC card session

Adapted from (Zenebe & Miao, 2001)
STRENGTHS

• The cards and the exercise are non-threatening & informal
• Provide a good environment for working and learning.
• Inexpensive, portable, flexible, and readily available
• Allow the participants to experience first hand how the system will work
• Useful tool for teaching people the object-oriented paradigm

Adapted from (Zenebe & Miao, 2001)
LIMITATIONS

• Provide only limited help in the aspects of design.
• Do not have enough notational power to document all the necessary components of a system.
• Do not specify implementation specifics.
• Can not provide view of the states through which objects transition during their life cycle.

Adapted from (Zenebe & Miao, 2001)
CRC GOOD PRACTICE

• Start with the simplest scenarios.
• Take the time to select meaningful class names.
• Take the time to write a description of the class.
• If in doubt, act it out!
• Lay out the cards on the table to get an intuitive feel for system structure.
• Be prepared to be flexible.

Adapted from (Zenebe & Miao, 2001)
Case Example:
A small technical library system for an R&D organization

- Requirement Statement
- Participants (Who? Why?)
- Creating Classes
- The CRC Card Sessions
  - scenario execution

Adapted from (Zenebe & Miao, 2001)
Case example: *Finding Classes*

- **Suggested Classes**
  - Library, Librarian, User, Borrower, Article, Material, Item, Due Date, Fine, Lendable, Book, Video, and Journal

- **Classes after filtering**
  - Librarian, Lendable, Book, Video, Journal, Date, Borrower and User

- **Assigning Cards**
  - A CRC Card per Class, put name & description of the class

Adapted from (Zenebe & Miao, 2001)
Scenario execution

- Scenario executions/Role Plays (For what?)
  - Filter and test identified classes
  - Identify additional classes
  - Identify responsibilities and collaborators
    - can be derived from the requirements/use cases
    - responsibilities that are "obvious" from the name of the class (be cautious, avoid extraneous responsibilities)
  - Filter and test responsibilities and collaborators
  - Attributes (only the primary ones)

Adapted from (Zenebe & Miao, 2001)
Finding Responsibilities

• Things that the class has knowledge about, or things that the class can do with the knowledge it has

• Tips/Indicators
  – Verb phrases in the problem or use case
  – Ask what the class knows? What/how the class does?
  – Ask what information must be stored about the class to make it unique?

Adapted from (Zenebe & Miao, 2001)
Finding Collaborators

• A class asks another class when it
  – needs information that it does not have or
  – needs to modify information that it does not have

• Client - Server relationship

• Tips/Indicators
  – Ask what the class does not know and needs to
    know? And who can provide that
Case example: *Scenario Execution*

- Identify Scenarios (By domain experts)
- Main scenarios: check-out, return and search
- Start with the simple ones
- The first one always takes the longest
- Domain experts have high level of contribution during the early scenarios

Adapted from (Zenebe & Miao, 2001)
Case example: *Checkout Scenario*

- Who should have the overall responsibilities for the task/check out? *Librarian*.
- What does the task entail?
- Shouldn't there be collaborations in the opposite direction?
  - Collaborations in CRC cards are one-way relationships from the client to the server (OO)
- Who should do the checking out of the Book? Librarian or Book itself? (Controversial)

Adapted from (Zenebe & Miao, 2001)
Case example: *Checkout Scenario*

- Who should tell Borrower to update its knowledge about outstanding Book? Librarian or Book?
- Do we need a collaboration between Book and Borrower for the “know set of books” responsibility?
  - Collaborations are not usually needed for responsibilities that simply hold information, only for situations where an object actually *sends a message* to a Collaborator.
  - Borrower does not need Book's help to put a Book in a set.

Adapted from (Zenebe & Miao, 2001)
CRC Cards after the first scenario run

<table>
<thead>
<tr>
<th>Librarian</th>
<th>Borrower, Book</th>
</tr>
</thead>
<tbody>
<tr>
<td>check out book for User</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>compare dates</td>
<td>Date</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Borrower</th>
<th>Book</th>
</tr>
</thead>
<tbody>
<tr>
<td>can borrow</td>
<td></td>
</tr>
<tr>
<td>know set of books</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Book</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>know if overdue</td>
<td></td>
</tr>
<tr>
<td>check out</td>
<td></td>
</tr>
<tr>
<td>calculate due date</td>
<td></td>
</tr>
<tr>
<td>know due date</td>
<td></td>
</tr>
<tr>
<td>know borrower</td>
<td></td>
</tr>
<tr>
<td>know in or out</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.3 Library application cards after the first scenario.

Adapted from (Zenebe & Miao, 2001)
Case example: *Search Scenario*

- "What happens when Ned comes to Library in search of a book entitled The Mythical Mammoth?"

- Discovery of new class: Collection class (Why?)
  - Book can’t look for itself
  - Collection looks over a set of Books to find the correct one

- When to end scenario execution?
  - When you have a stable model (does not cause new C or R to be added)

Adapted from (Zenebe & Miao, 2001)
Grouping Cards

- CRC cards on the table provides a visual representation of the emerging model
- Classes with hierarchical (is-a) relationship
- Class who collaborate heavily placed closer
- Class included by other class (has-a relationship); e.g. Date in Lendable
- Card clustering based on heavy usage or collaborations can provide visual clues to subsystems

Adapted from (Zenebe & Miao, 2001)
Lower-Level Design

• CRC cards can be used to:
  – continually refine the classes
  – add implementation details
  – add classes not visible to user, but to designers and programmers
  – add classes needed for implementation, e.g.
    • Database
    • User Interface
    • Error Handling

Adapted from (Zenebe & Miao, 2001)
Lower-Level Design

• Considering Design Constraints
  – Choice of supporting software components
  – Target environment and language
  – Performance requirements: response-time/speed, expected availability, number of users
  – Errors/exceptional handling
  – Others: Security, Memory, etc.
Lower-Level Design

• “Design Classes”
  – represent mechanisms that support implementation of the problem
  – contain the data structures and operations used to implement the user-visible classes e.g. Array, List
  – interface classes for UI and DBM subsystems
  – classes to handle error conditions
Lower-Level Design

• Important questions:
  • Who creates this object?
  • What happens when it is created and adopted?
  • What is the lifetime of the object vs. the life time of the information (persistence) held by an object?

• Attributes
  • Discovery of attributes that are necessary to support the task during examination of each responsibility
  • Identification of persistent attributes
  • Leads to a database design (database model)

Adapted from (Zenebe & Miao, 2001)
Case example: *Lower-level Design*

- Brainstorming any classes that come to mind based on design constraints such as
  - User Interface, Database access, error handling
  - User Interact class & DB interface Classes

- Scenario identification and execution
  - Object creation scenarios
  - Check-out Scenario
  - Return Scenario
  - Search Scenario

- **Output:** Design classes

Adapted from (Zenebe & Miao, 2001)
Case example: *Lower-level Design*

- **Principles:**
  - make independent of specific hardware and software products
  - use specific class names instead of generic names such as GUI and DBMS
  - Work on both normal and exceptional scenarios

Adapted from (Zenebe & Miao, 2001)
Case example: *Lower-level Design*

- New classes identified:
  - **User interface:** to get input from and output to user *using* GUI software classes
  - **Database:** To obtain and store Borrower objects and objects of the Lendable classes *using* DBMS software classes

Adapted from (Zenebe & Miao, 2001)
Deliverables

• Complete list of CRC Cards (class descriptions)
• List of scenarios recorded as suggested and executed
• Collaboration Diagram
• Application Problem Model
Advantages of CRC Cards

• Common project vocabulary
• Spreading domain knowledge
• Spreading OO design expertise
• Implicit design reviews
• Live prototyping
• Identifying holes in the requirements
• Limitation: Informal notation
  – “Designing is not the act of drawing a diagram” (Booch)

Adapted from (Zenebe & Miao, 2001)