Determining the Key Abstractions

Objectives

Upon completion of this module, you should be able to:

- Identify a set of candidate key abstractions
- Identify the key abstractions using CRC analysis

Additional Resources

Additional resources – The following references provide additional information on the topics described in this module:

- Beck, Kent, Ward Cunningham. *A Laboratory For Teaching Object-Oriented Thinking*, [`http://c2.com/doc/oopsla89/paper.html`] 1989.
- Booch, Grady. *Object-Oriented Analysis and Design with Applications (2nd ed).* The Benjamin/Cummins Publishing Company, Inc., Redwood City, 1994.
- Folwer, Martin, Kendall Scott. *UML Distilled (2nd ed)*. Reading: Addison Wesley Longman, Inc., 2000.

Process Map

This module covers the next step in the Requirements Analysis workflow: discovering key abstractions. Figure 6-1 shows the activities and artifacts covered in this module.

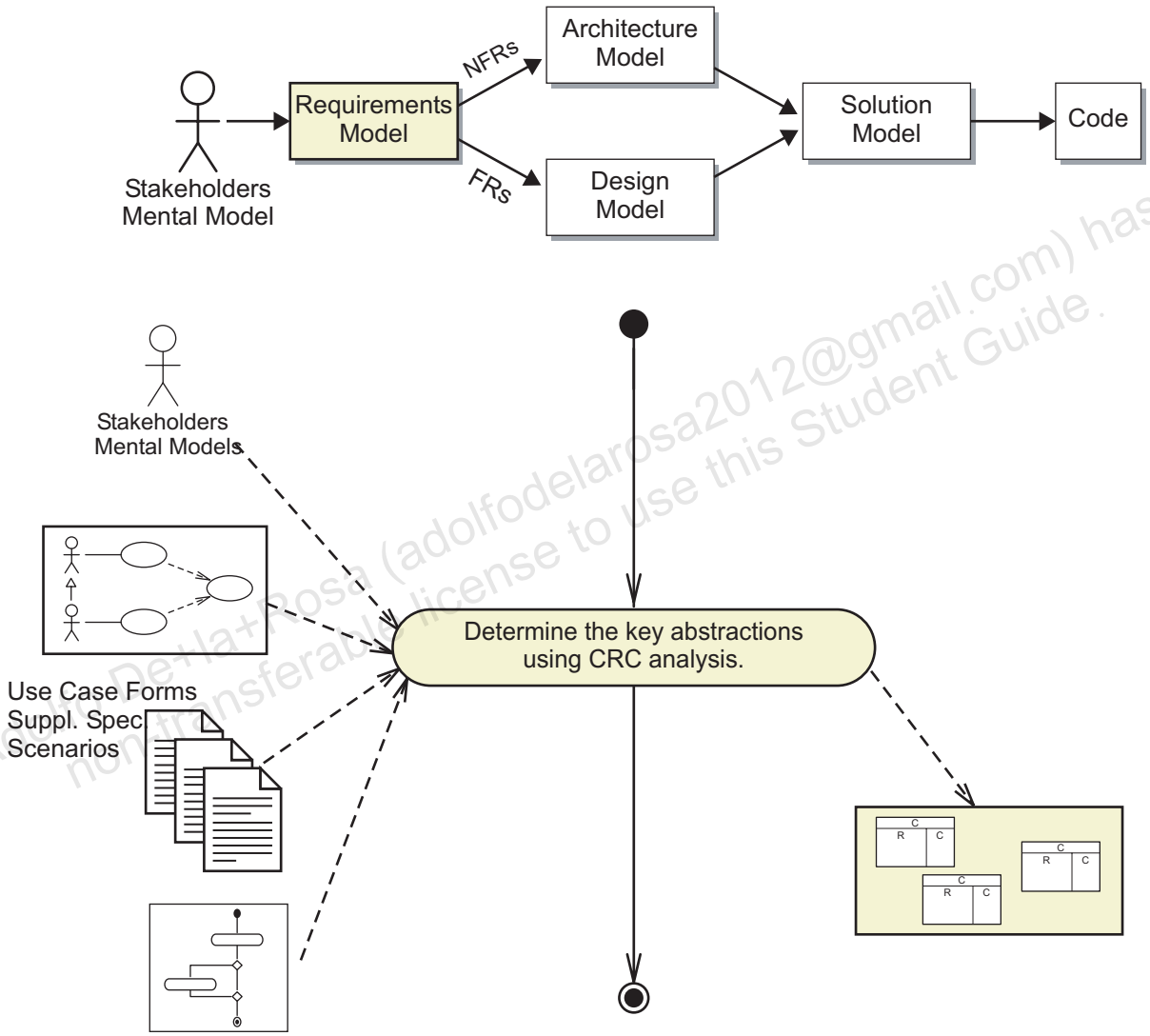

Figure 6-1 Key Abstractions Process Map

Introducing Key Abstractions

> "A key abstraction is a class or object that forms part of the vocabulary of the problem domain." (Booch, page 162)

Key abstraction is the name for the primary objects within the system. These represent an integral part of the language in which the domain experts and the end users communicate about the business.

Determining the key abstractions for a domain is a process of discovery:

1. Identify all candidate key abstractions by listing all nouns from the project artifacts in a "Candidate Key Abstractions Form."
2. Use CRC analysis to determine the essential set of key abstractions.

   Key abstractions are recognized as objects that have responsibilities and are used by other objects (the collaborators).

This module describes only one technique for discovering key abstractions. There are other ways to manage this process, but it is beyond the scope of the course to cover all these techniques.

Identifying Candidate Key Abstractions

Begin the process of identifying all of the unique nouns in the project artifacts by focusing on the following areas of the documents:

- The Main Flow and Alternate Flow sections of the use case forms.

  The nouns in these sections are usually essential to understand the problem domain. This is where you will find details and responsibilities of the problem domain.
- The other sections of the use case forms
- The use case scenarios
- The Glossary of Terms
- The Supplementary Specification Document.

  For large projects that use a non-iterative/incremental development process, this task would be time-consuming.

  For an iterative/incremental development process you will only need the current artifacts or changes to existing artifacts to be considered.

With practice you will be able to skip some of the nouns that are obviously not part of the domain.

Identifying the Candidate Abstractions

An easy technique to scan the Use Case Form for nouns is to print a copy, and use a highlighter pen to mark each significant noun.

Here are a few excerpts from the Hotel System artifacts, with the nouns marked in **bold**:

- From the Create Reservation Use Case Form Description section:

  The **Customer** requests a **reservation** for **hotel rooms** for a **date range**. If all the requested **rooms** are available, the **price** is calculated and offered to the **Customer**. If **details of the customer** and a **payment guarantee** are provided, the **reservation** will be confirmed to the **Customer**.

- From the Create Reservation Use Case Form Main Flow section:

  1: **Use case** starts when **Customer** requests to create a **reservation**
  2: **Customer** enters **types of rooms**, **arrival date**, and **departure date** [A1] [A2]
  2.1: **Systems** creates a **reservation** and reserves **rooms** applying BR3 [A3]
  2.3: **System** calculates **quoted price** applying BR4
  2.3.1 **System** records **quoted price**
  2.4: **System** notifies **Customer** of **reservation details** (including **rooms** and **price**)
  3: **Customer** accepts **rooms** offered [A5]
  3.1: Extension Point (new **customer**) [A6]
  3.2: Extension Point (**payment guarantee**) [A7]
  3.3: **System** changes **reservation status** to "confirmed"
  3.4: **System** notifies **Customer** of **confirmed reservation** details
  4: Use case ends

- From the Create Reservation Use Case Form Alternate Flow section:

  A1: **Customer** can enter **duration** instead of **departure date**, go to step 2.1 [A2]
  A2: Failed date check BR1. Notify **error** to **Customer**, go to step 2
  A3: Complying with BR2, **System** determines that required **rooms** are not available, **System** upgrades one or more **room types**, go to step 2.1 [A4]
  A4: No further upgrades available. Notify **message** to **Customer**, go to step 2
  A5: **Rooms** offered are declined, go to step A9
  A6: **Customer** already exists, **Customer** enters **customer name** and **zip code**, **System** searches for matching **customers**, notifies

Customer of matching **customers**, **Customer** selects correct **customer**, go to step 3.2 [A8]
A7: **Payment guarantee** fails. Notify **message** to **Customer**, go to step 3.2
A8: Existing **customer** not found, go to step 3.1
A9: **Reservation** not confirmed, **reservation** deleted, use case ends
At any time: **Customer** may cancel the use case, use case ends [A9]
After use case inactivity of 10 minutes: use case ends [A9]

- From the Create Reservation Use Case Form Business Rules section:

  BR1: The **arrival date** must not be before **today's date**, and the **departure date** must be after the **arrival date**
  ...
  BR3: **Reservations** with assigned **rooms** but no **payment guarantee** have a **status** of "held"
  ...
  BR8: **Reservations** with a **status** of "confirmed" must be linked to a **payment guarantee** and a **customer**
  BR9: **Reservation** must not exist without being linked to at least one **room**

- From the Create Reservation Use Case Form Remaining Sections:
  ...

- From the Supplementary Specification Documents. For example the Project Glossary, see Table 6-1

Table 6-1 Project Glossary Terms

Term	Definition
Reservation	An allocation of a specific **number of rooms**, each of a specified **room type**, for a specified **period of days**.
Date Range	Specifies a **start date** and an **end date**

Note – We have deliberately omitted some text with important nouns. This will enable you to find them, as part of your lab.

Candidate Key Abstractions Form

The form for recording candidate key abstractions uses three fields:

- Candidate key abstraction

  This field contains a noun discovered from the project analysis artifacts.

- *Reason for Elimination*

  This field is left blank if the candidate becomes a key abstraction. Otherwise, this field contains the reason why the candidate was rejected.

- *Selected name*

  Ultimately, a key abstraction will become some sort of software component, such as a class. This field contains the name of the class if this entry is selected as a key abstraction.

Use this form to record all potential (candidate) key abstractions from the set of nouns in the project analysis artifacts. You will list the nouns in the first column. During the CRC analysis (described in the "Discovering Key Abstractions Using CRC Analysis" on page 6-11) you will fill in the rest of the columns.

Table 6-2 shows an excerpt from the Candidate Key Abstractions Form for the Hotel System.

Table 6-2 Initial Candidate Key Abstractions Form for the Hotel Reservation System

Candidate Key Abstraction	Reason for Elimination	Selected Component Name
Reservation		
Customer actor		
System		
Customer		
Room		
Date Range		
Price		
Customer Details		
Payment Guarantee		

Table 6-2 Initial Candidate Key Abstractions Form for the Hotel Reservation System

Candidate Key Abstraction	Reason for Elimination	Selected Component Name
Room Type		
Arrival Date		
Departure Date		
Quoted Price		
Reservation Details		
Reservation Status		
Confirmed Reservation		
Duration		
Customer Name		
Customer Zip Code		
Today's Date		
Period of Days		

Project Glossary

The process of identifying candidate key abstractions is also a good opportunity to verify that your project glossary is up-to-date.

- Verify that all domain-specific terms have been listed and defined.

  A candidate key abstraction is usually a domain-specific term. All of these should be listed in the glossary.

- Identify synonyms in the project glossary and select a primary term to use throughout the documentation and source code.

  Synonyms sometimes appear in the candidate key abstractions list. For example, a booking agent might refer "reservation" as "booking." Therefore, you should verify that both terms, "reservation" and "booking" appear in the glossary and that the definition of "booking" refers to the definition of "reservation."

Discovering Key Abstractions Using CRC Analysis

After you have a complete list of candidate key abstractions, you need to filter this list to determine the *essential set of key abstractions*. One technique is CRC (class-responsibility-collaboration) analysis.

To perform CRC analysis:

1. Select one candidate key abstraction.
2. Identify a use case in which this candidate is prominent.
3. Scan the use case forms and the use case scenarios that contain this noun to determine responsibilities and collaborators.
4. Scan the Glossary of Terms for all references to the noun.
5. Document this key abstraction with a CRC card.
6. Update Candidate Key Abstractions Form based on findings.

This process is iterative. You will start with one key abstraction candidate and then evaluate whether the candidate is a true key abstraction. When you have updated the form (Step 5), you will then select another candidate from the list and continue this process until all candidates in the form have been considered.

Each step in this process is described in the following sections.

Note – CRC analysis was invented by Ward Cunningham and Kent Beck in the late 80s to help teach object-oriented design. For more information about CRC analysis visit the URL `http://c2.com/doc/oopsla89/paper.html`.

Selecting a Key Abstraction Candidate

Selecting a good key abstraction candidate is largely intuition, but here are a few tactics:

- Ask a domain expert.

  It is useful to have a domain expert in the meeting during CRC analysis. This person can immediately tell the analysis team if a candidate is likely to be a real key abstraction. Their mental model can quickly distinguish an object from an attribute, subtype name, or operation name.

If you do not have access to a domain expert, then the following tactic can be useful.

- Choose a candidate key abstraction that is used in a use case name.

  The use case names are usually written with a leading active verb (like "manage" or "create") followed by a noun (like "reservation"). The nouns in the use case names tend to be key abstractions.

- Choose a candidate key abstraction that is used in a use case form.

  These will contain most of the candidate key abstractions.

For the Hotel System, the noun "reservation" is a good candidate because it appears many times in the following areas of the project analysis:

- In these use case names:
  - Create *Reservation*
  - Update *Reservation*
  - Delete *Reservation*
- In many places throughout the use case forms. For example the

  Check In Customer use case form will describe assigning a Bill (Folio) to a *reservation*.

Identifying a Relevant Use Case

To determine whether the candidate key abstraction is a real key abstraction, you must determine if the candidate has any responsibilities and collaborators. To find the responsibilities and collaborators, you need to scan the text of the use case, details FRs for the use case, and the scenarios for the use case.

To identify a relevant use case that might declare a candidate's responsibilities and collaborators:

1. Scan the use case names for the candidate key abstraction.

   If the candidate key abstraction is mentioned in the name of a use case, it is very likely that the use case will be relevant.

2. Scan the use case forms for the candidate key abstraction.

   Likewise, if the candidate key abstraction is mentioned in the description of a use case, it is very likely that the use case will be relevant.

3. Scan the use case scenarios for the candidate key abstraction.

4. Scan the text of the use case scenarios to see if the candidate key abstraction is mentioned. If it is, the scenario will be relevant.

For the Hotel Reservation System, the noun "reservation" occurs in names of three use cases:

- Create *Reservation*
- Update *Reservation*
- Delete *Reservation*

These relevant use cases will be used to perform the next step in the CRC process – to discover the responsibilities and collaborations of the selected candidate key abstraction.

Determining Responsibilities and Collaborators

Scan the scenarios and use case forms of the identified use cases for responsibilities and collaborators of the candidate key abstraction. The responsibilities of a key abstraction are any attributes, operations, or specifications of the range of data value for attributes (for example, the names of the values of a state variable). The collaborators are other objects (usually another key abstraction) with which the candidate key abstraction is associated.

If you cannot find any responsibilities, then you can reject this candidate. Use the second column of the Candidate Key Abstractions Form to record the reason that the candidate was eliminated.

Also, if you determine that a responsibility (usually an attribute name or the name of a subtype) is also on the candidate list, then you can reject that noun from the list.

For example, the artifacts for the Hotel System includes the following details that specify a few of the responsibilities and collaborators of the Reservation key abstraction:

- Glossary Term Reservation has the following definition: an allocation of a specific number of rooms, each of a specified room type, for a specified period of days.

This glossary entry specifies that a reservation contains two important data items, the arrival and departure dates. These are responsibilities in the form of attributes. The glossary entry also specifies that a reservation is related to one or more rooms. This relationship is considered a collaboration.

- Business Rule BR9: Reservation must not exist without being linked to at least one room.

  This business rule entry specifies that a reservation must be linked to at least one room. This relationship confirms the collaboration between the Reservation and a Room.

- Business Rule BR8: Reservations with a status of "confirmed" must be linked to a payment guarantee and a customer

  This business rule entry specifies that a reservation contains another important data item: the status of the reservation. This is a responsibility in the form of an attribute. The business rule also specifies that a reservation is linked to a payment guarantee and a customer. These relationships are considered as collaborations.

- Main Flow 3.3: System changes reservation status to "confirmed"

  This main flow entry specifies confirms that a reservation contains the important data item: the status of the reservation. This is a responsibility in the form of an attribute.

Note that the analysis for the Reservation key abstraction has identified attributes (for example, arrival date) and collaborators. The attributes might also be on the candidate key abstractions list; these can be eliminated with the reason that they are attributes.

Documenting a Key Abstraction Using a CRC Card

After a key abstraction has been identified, you will create a CRC card to record the responsibilities and collaborators of this key abstraction. Figure 6-2 shows a template of a CRC card.

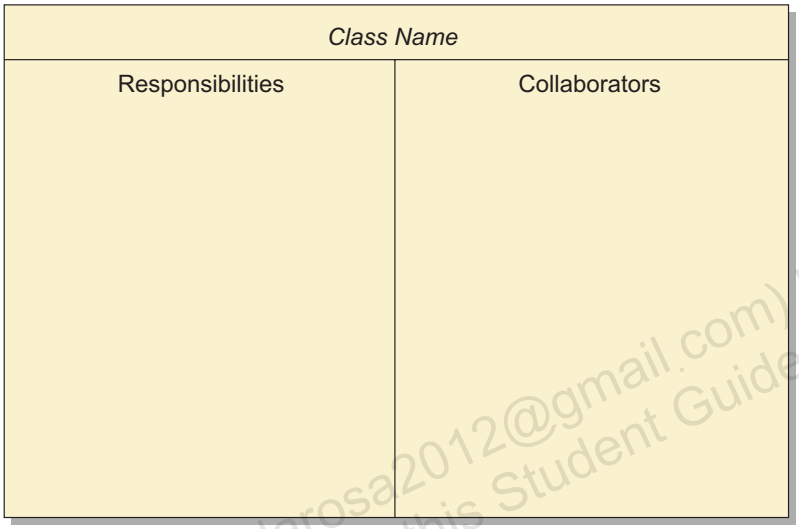

Figure 6-2 A CRC Card Template

The class name field at the top of the CRC card is where you place the name of the key abstraction. Finding a good name is important, especially if there are synonyms for this concept. As a recommendation, keep the name simple and short, but do not use abbreviations unless the full name is too long.

Use the responsibility column to record the operations and attributes of the key abstraction. Use the collaborators column to record any relationships and behavioral collaborations for the key abstraction.

Figure 6-3 shows an example CRC card for the Reservation key abstraction.

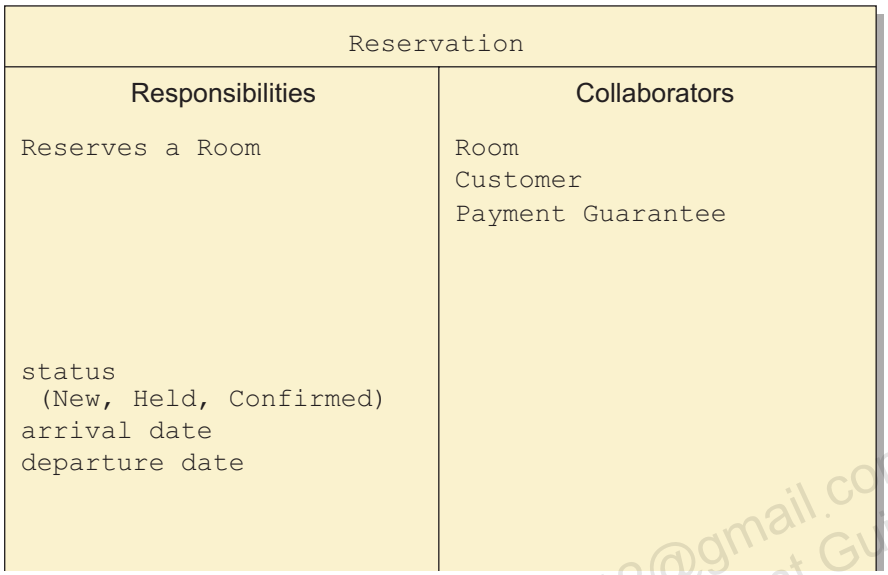

Figure 6-3 The CRC Card for the Reservation Key Abstraction

These cards are more than just a source of documentation. Use them as *role-playing exercises* that help the analysis team and the domain expert explore how each key abstraction is used by the system to accomplish the use cases. It is this role-playing of the use case scenarios that is essential to the CRC process.

For example, consider the Create Reservation use case scenario 1. One member of the analysis team would represent the reservation object (holding the Reservation CRC card) and another member of the team would represent the customer object (holding the Customer CRC card). A third member of the team would represent the Booking Agent actor. This person would interact with the two other players as the scenario specifies activities designated for the specific object; for example, adding the customer to the reservation.

The role-playing exercise is an informal process, but it can be a powerful technique for building a deeper understanding of the problem domain.

Updating the Candidate Key Abstractions Form

If the candidate you selected has responsibilities, then enter the name of the key abstraction (from the CRC card) into the “Selected Name” field.

If the candidate you selected has responsibilities and no collaborators, then this candidate must be rejected. Enter an explanation of why the candidate was not selected as a key abstraction. Some reasons for rejecting a candidate include:

- Name of an attribute
- Name of a subtype
- Name of an external system or any actor
- Name of a value of an attribute (such as for a status attribute)

After several iterations of the CRC process, several key abstractions are discovered for the Hotel System. Many candidates were rejected. Table 6-3 shows an excerpt of the final Candidate Key Abstractions Form.

Table 6-3 Final Candidate Key Abstractions Form for the Hotel Reservation System

Candidate Key Abstraction	Reason for Elimination	Selected Name
Reservation		Reservation
Customer actor	External to system	
System	The whole system	
Customer		Customer
Room		Room
Date Range	Synonym for Arr. and Dept. Date	
Price	Synonym for Quoted Price	
Customer Details	Same as Customer	
Payment Guarantee		Payment Guarantee
Room Type		RoomType
Arrival Date	Attribute of Reservation	
Departure Date	Attribute of Reservation	
Quoted Price	Attribute of Reservation	
Reservation Details	Same as Reservation	
Reservation Status	Attribute of Reservation	
Confirmed Reservation	Type of Reservation	

Table 6-3 Final Candidate Key Abstractions Form for the Hotel Reservation System

Candidate Key Abstraction	Reason for Elimination	Selected Name
Duration	Derived from Arr. and Dept. Date	
Customer Name	Attribute of Customer	
Customer Zip Code	Attribute of Customer	
Today's Date	External to System	
Period of Days	Synonym for Duration	

Summary

In this module, you were introduced to the process of discovering the Key Abstraction of the problem domain. Here are a few important concepts:

- Key abstractions are the essential nouns in the language of the problem domain.
- To identify the key abstractions:
  a. List all (problem domain) nouns from the project analysis artifacts in a Candidate Key Abstractions Form.
  b. Use CRC analysis to identify the key abstractions (a class with responsibilities and collaborators) from the candidate key abstractions list.