Lecture 7

Discovering Class Responsibilities and Collaborators with CRC Cards

Philipp Reinecke
reineckep@cardiff.ac.uk

Objects Collaborate

- The responsibilities of a class are carried out by its operations and attributes.

- No object can carry out every responsibility on its own. The objects must collaborate to bring about the desired behaviour in the system.

Class-Responsibility-Collaboration - CRC

- Class – an entity that will exist in the system you are developing (who)
- Responsibility – a high level description of the purpose of the class (does what)
  - What the object knows
  - What the object does
- Collaboration – other classes the class needs to work with to fulfil the responsibility (with whom)

CRC Cards

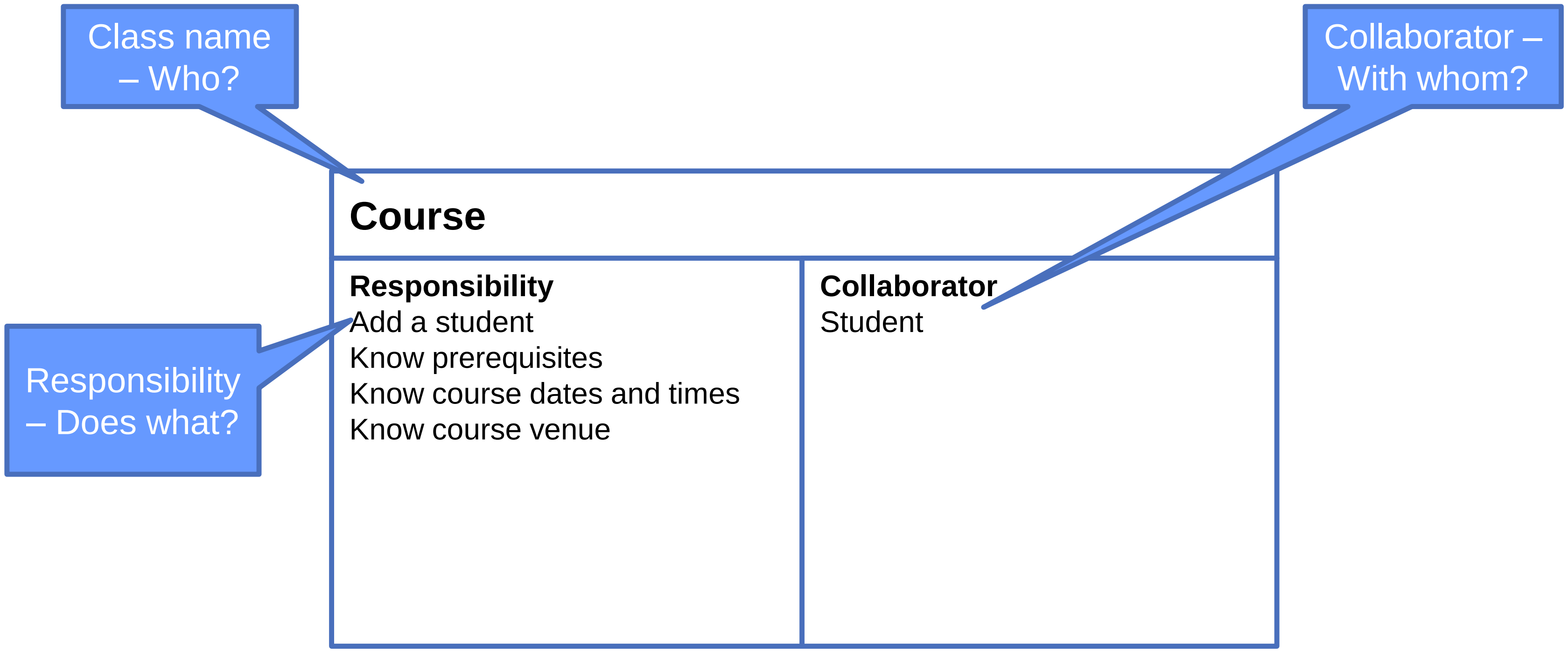
- You can discover responsibilities with Class Responsibility Collaboration (CRC) Cards.
  - developed by Kent Beck and Ward Cunningham in the late 1980's.

- A CRC card is a 4"X6" index card that contains the:
  - class name and description.
  - responsibilities of the class.
  - collaborators for the responsibilities.

CRC card

- A CRC card captures:

  Who - Does What - With Whom



What is a CRC Session?

A CRC session is run to help discover the class responsibilities and collaborations necessary for a use case to be realized.

Begins by selecting one scenario from the use case.

1. Participants break up into teams of two.
2. Each team is assigned specific classes that it will represent in the session.
3. The session leader acts in the role of the Actor that initiates the use case.
4. The actor will “go” to a specific class to begin the interaction.

Example: Close Registration Use Case

Close Registration

Brief Description

This use case allows a Registrar to close the registration process. Course offerings that do not have enough students are cancelled. Course offerings must have a minimum of three students in them. The billing system is notified for each student in each course offering that is not cancelled, so the student can be billed for the course offering.

Close Registration CRC Session

This use case starts when the Registrar requests that the system close registration.

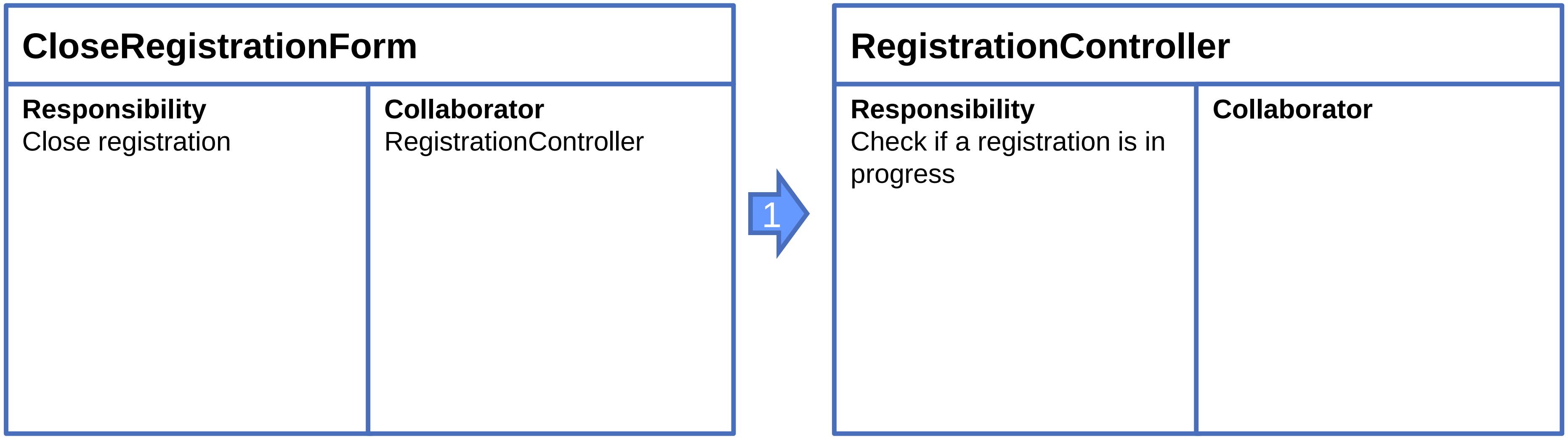
- An actor can only interact with the system boundary. Create a CloseRegistrationForm class to model this.
- Assign a responsibility *Close registration* to this class.
- To fulfil this responsibility, CloseRegistrationForm needs to delegate this responsibility to a control class that will be responsible for controlling the logic of the use case.
- Create RegistrationController class. It will collaborate with the Form to fulfil the Close registration responsibility.

CloseRegistrationForm	
Responsibility	**Collaborator**
Close registration	RegistrationController

Close Registration CRC Session

- RegistrationController class collaborates with the Form to fulfil the Close registration responsibility.

***The system checks to see if a registration is in progress**. If it is, then a message is displayed to the Registrar and the use case terminates. The Close registration processing cannot be performed if a registration is in progress*.



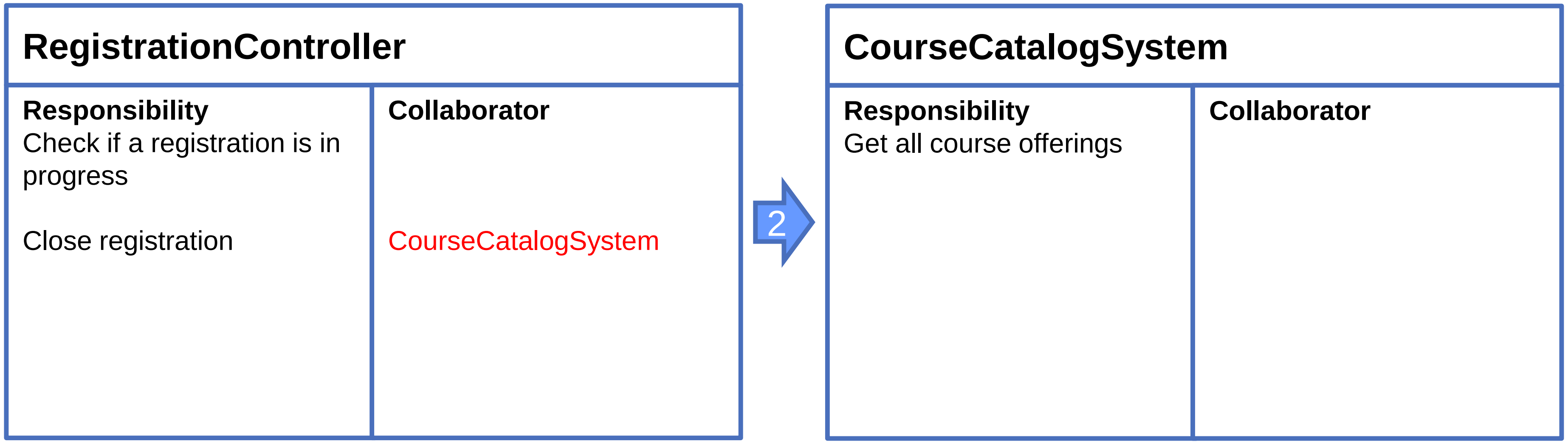
CloseRegistrationForm	
Responsibility	**Collaborator**
Close registration	RegistrationController

RegistrationController	
Responsibility	**Collaborator**
Check if a registration is in progress	

Close Registration CRC Session cont.

If registration is closed, then this use case resumes the close the registration process.

- To Close registration, RegistrationController needs to get all the course offerings.
- CourseCatalogSystem has the information about course offerings.
- Controller class collaborates with CourseCatalogSystem to *Get all course offerings.*
- CourseCatalogSystem can do this responsibility by itself and does not need any collaborators.



RegistrationController	
Responsibility	**Collaborator**
Check if a registration is in progress	
Close registration	CourseCatalogSystem

CourseCatalogSystem	
Responsibility	**Collaborator**
Get all course offerings	

Close Registration CRC Session cont.

Use Case Step 2

For each course offering, the system checks if a professor has signed up to teach the course offering.

- CourseOffering class has this information.
- To do this, the RegistrationController needs to collaborate with the CourseOffering class.
- CourseOffering gets the responsibility *Close course offering.* This responsibility will determine if an instructor has signed up for the course and, if so, to mark it for closing.

RegistrationController	
Responsibility	**Collaborator**
Check if a registration is in progress	
Close registration	CourseCatalogSystem
	CourseOffering

3

CourseOffering	
Responsibility	**Collaborator**
Close course offering	

Close Registration CRC Session cont.

Use Case Step 2

The system must also commit the course offering for each schedule that contains it.

- This is still part of the *Close course offering* responsibility.

- To mark the schedule for closing, CourseOffering must collaborate with Schedule to complete its *Close course offering* responsibility.

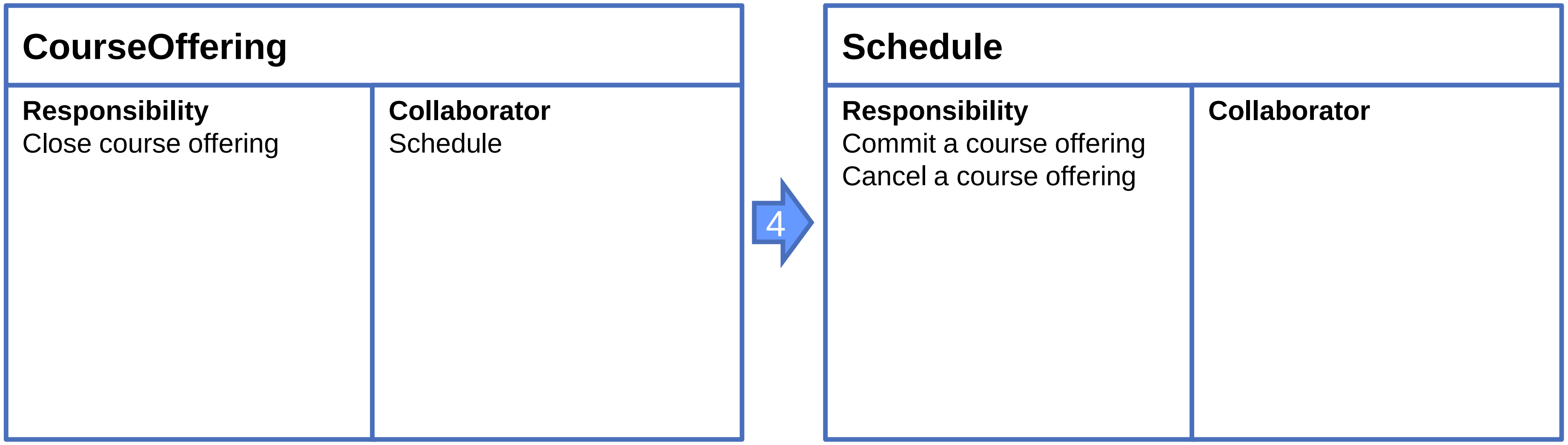
CourseOffering	
Responsibility	**Collaborator**
Close course offering	Schedule

Close Registration CRC Session cont.

Use Case Step 2

Whether a course offering can or cannot be closed, the Schedule needs to be notified.

- Hence, there are two responsibilities assigned to Schedule, *Commit a course offering* and *Cancel a course offering*, that account for these two situations.
- At this stage, use case control returns back to the RegistrationController.



CourseOffering	
Responsibility	**Collaborator**
Close course offering	Schedule

Schedule	
Responsibility	**Collaborator**
Commit a course offering	
Cancel a course offering	

Close Registration CRC Session cont.

Use Case Step 3

For each schedule, the system "levels" the schedule: if the schedule does not have the maximum number of primary courses selected, the system attempts to select alternates from the schedule's list of alternates. The first available alternate course offerings will be selected. If no alternates are available, then no substitution will be made.

- The RegistrationController must collaborate with the Schedule class to perform the responsibility: Level the Schedule

RegistrationController

Responsibility	Collaborator
Check if a registration is in progress	
Close registration	CourseCatalogSystem CourseOffering Schedule

5

Schedule

Responsibility	Collaborator
Commit a course offering Cancel a course offering Level schedule	

Close Registration CRC Session cont.

Use Case Step 4

For each course offering, the system closes all course offerings. If the course offerings do not have at least three students at this point (some may have been added as a result of levelling), then the system cancels the course offering. The system cancels the course offering for each schedule that contains it.

- This step is already captured within the *Close course offering* responsibility of the CourseOffering

CourseOffering	
Responsibility	**Collaborator**
Close course offering	Schedule

Close Registration CRC Session cont.

Use Case Step 5

The system calculates the tuition owed by each student for his current semester schedule.

- RegistrationController collaborates with Student to do this.
- Student get the responsibility Calculate tuition.

RegistrationController	
Responsibility	**Collaborator**
Check if a registration is in progress	
Close registration	CourseCatalogSystem CourseOffering Schedule Student

6

Student	
Responsibility	**Collaborator**
Calculate tuition	

Close Registration CRC Session cont.

Use Case Step 5

The system calculates the tuition owed by each student for his current semester schedule.

- The Student class needs to collaborate with the Schedule class to get the cost of his/her schedule.
- Schedule class get the responsibility Get cost

Student	
Responsibility Calculate tuition	**Collaborator** Schedule

Schedule	
Responsibility Commit a course offering Cancel a course offering Level schedule Get Cost	**Collaborator**

Close Registration CRC Session cont.

Use Case Step 5

The system calculates the tuition owed by each student for his current semester schedule and sends a transaction to the Billing System. The Billing System will send the bill to the students, which will include a copy of their final schedule.

- The RegistrationController needs to collaborate with the BillingSystem class to do this.

RegistrationController

Responsibility	Collaborator
Check if a registration is in progress	
Close registration	CourseCatalogSystem CourseOffering Schedule Student BillingSystem

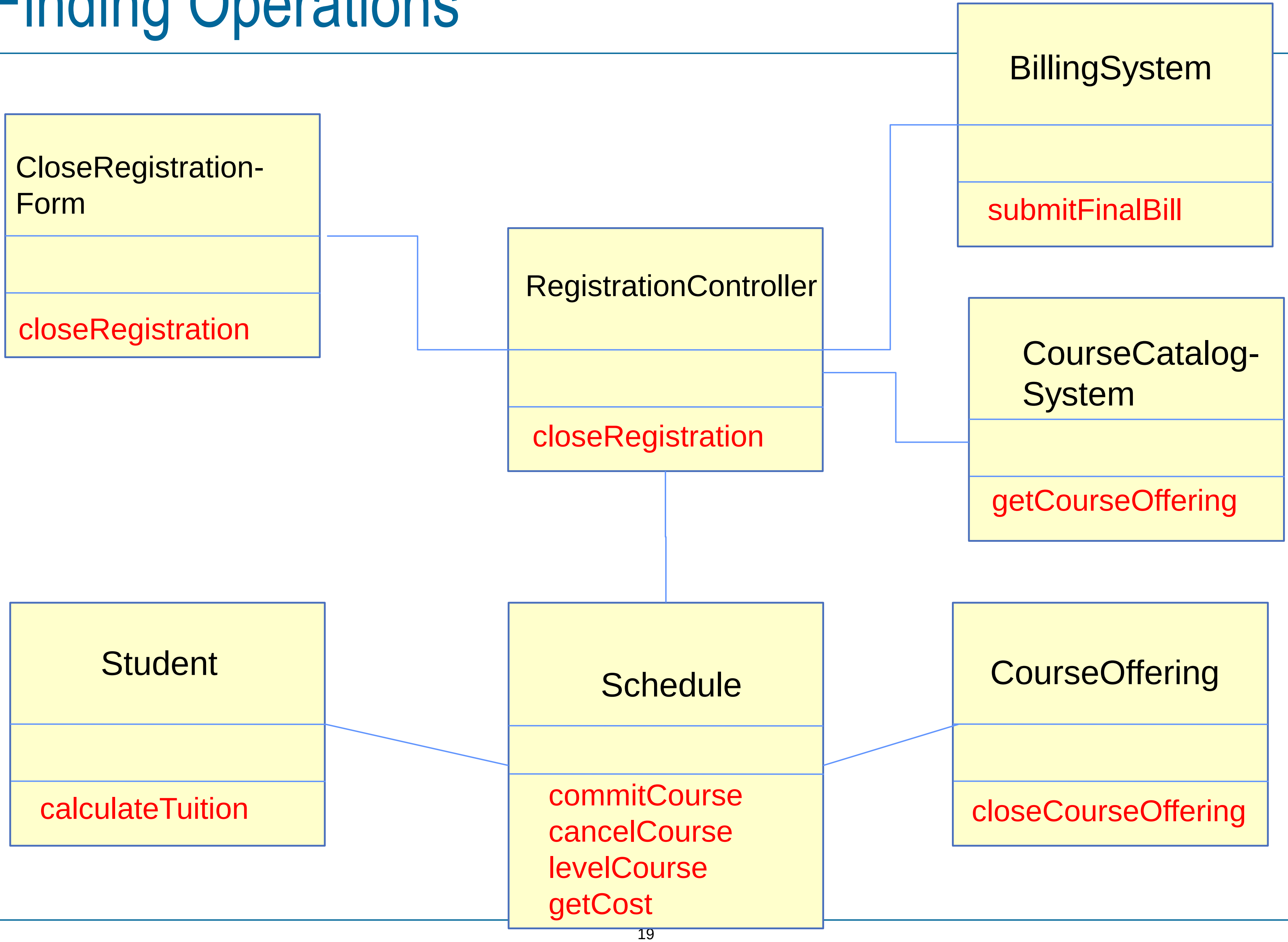
8

BillingSystem

Responsibility	Collaborator
Submit final bill	

The basic flow has been carried out and the session is over.

Finding Operations



Summary

- Objects are useless unless they can work together to solve a problem.
- Objects "advertise" their responsibilities through their operations.
- Class responsibilities are best discovered by "becoming the object." CRC cards facilitate this process.
- One of the most important benefits of CRC cards is that collaboration patterns begin to emerge.
- A CRC session is run to help discover the class responsibilities and collaborations necessary for a use case to be realized.

Modelling Exercise

Dashing through the snow
In a one-horse open sleigh
O'er the fields we go
Laughing all the way

Bells on bob tail ring
Making spirits bright
What fun it is to ride and sing
A sleighing song tonight!

Jingle bells, jingle bells,
Jingle all the way.
Oh! what fun it is to ride
In a one-horse open sleigh.

Jingle bells, jingle bells,
Jingle all the way;
Oh! what fun it is to ride
In a one-horse open sleigh.

Any questions?