agilemodeling.com

Class Responsibility Collaborator (CRC) Models: An Agile Introduction

11-13 minutes

A Class Responsibility Collaborator (CRC) model (Beck & Cunningham 1989; Wilkinson 1995; Ambler 1995) is a collection of standard index cards that have been divided into three sections, as depicted in Figure 1. A class represents a collection of similar objects, a responsibility is something that a class knows or does, and a collaborator is another class that a class interacts with to fulfill its responsibilities. Figure 2 presents an example of two hand-drawn CRC cards.

Figure 1. CRC Card Layout.

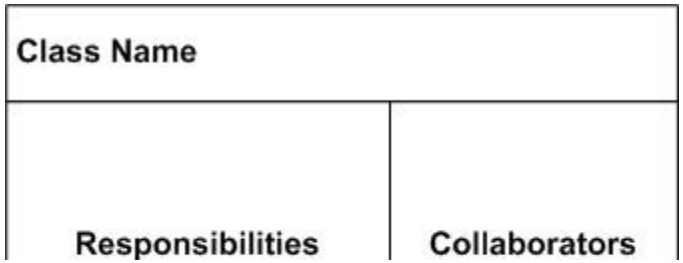
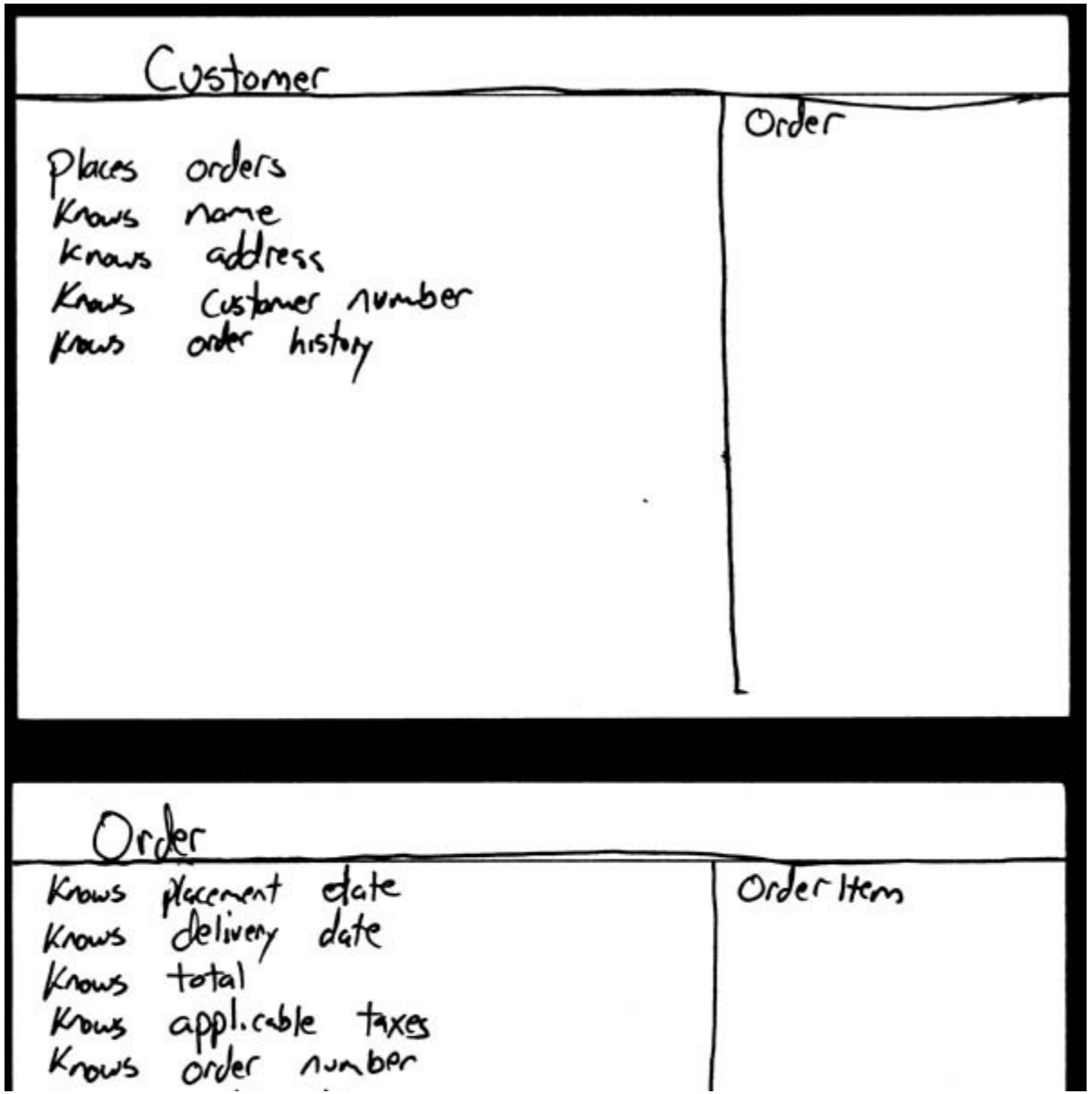


Figure 2. Hand-drawn CRC Cards.



Although CRC cards were originally introduced as a technique for teaching object-oriented concepts, they have also been successfully used as a full-fledged modeling technique. My experience is that CRC models are an incredibly effective tool for conceptual modeling as well as for detailed design. CRC cards feature prominently in eXtreme Programming (XP) (Beck 2000) as a design technique. My focus here is on applying CRC cards for conceptual modeling with your stakeholders.

A class represents a collection of similar objects. An object is a person, place, thing, event, or concept that is relevant to the system at hand. For example, in a university system, classes would represent students, tenured professors, and seminars. The name of the class appears across the top of a CRC card and is typically a singular noun or singular noun phrase, such as *Student*, *Professor*,

and *Seminar*. You use singular names because each class represents a generalized version of a singular object. Although there may be the student John O'Brien, you would model the class *Student*. The information about a student describes a single person, not a group of people. Therefore, it makes sense to use the name *Student* and not *Students*. Class names should also be simple. For example, which name is better: *Student* or *Person who takes seminars*?

A responsibility is anything that a class knows or does. For example, students have names, addresses, and phone numbers. These are the things a student knows. Students also enroll in seminars, drop seminars, and request transcripts. These are the things a student does. The things a class knows and does constitute its responsibilities. Important: A class is able to change the values of the things it knows, but it is unable to change the values of what other classes know.

Sometimes a class has a responsibility to fulfill, but not have enough information to do it. For example, as you see in Figure 3 students enroll in seminars. To do this, a student needs to know if a spot is available in the seminar and, if so, he then needs to be added to the seminar. However, students only have information

about themselves (their names and so forth), and not about seminars. What the student needs to do is collaborate/interact with the card labeled *Seminar* to sign up for a seminar. Therefore, *Seminar* is included in the list of collaborators of *Student*.

Figure 3. Student CRC card.

Student	
Student number Name Address Phone number Enroll in a seminar Drop a seminar Request transcripts	Seminar

Collaboration takes one of two forms: A request for information or a request to do something. For example, the card *Student* requests an indication from the card *Seminar* whether a space is available, a request for information. *Student* then requests to be added to the *Seminar*, a request to do something. Another way to perform this logic, however, would have been to have *Student* simply request *Seminar* to enroll himself into itself. Then have *Seminar* do the work of determining if a seat is available and, if so, then enrolling the student and, if not, then informing the student that he was not enrolled.

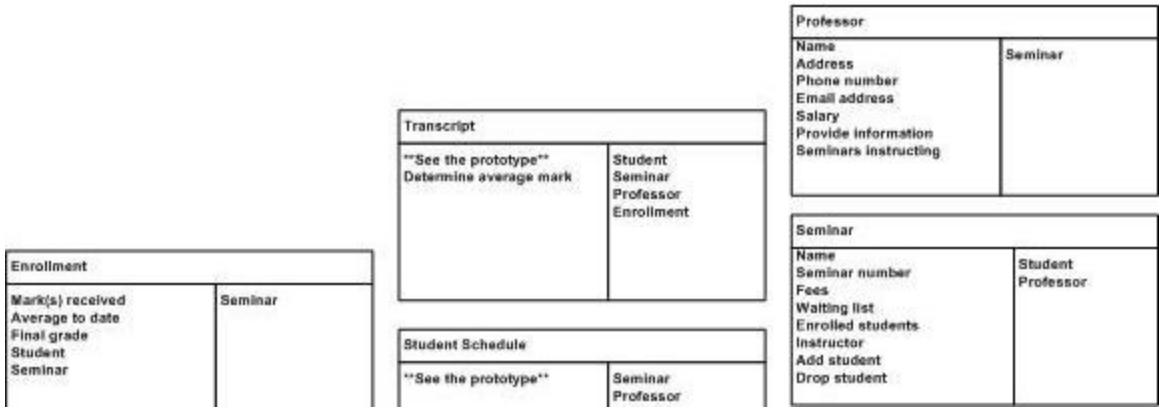
So how do you create CRC models? Iteratively perform the following steps:

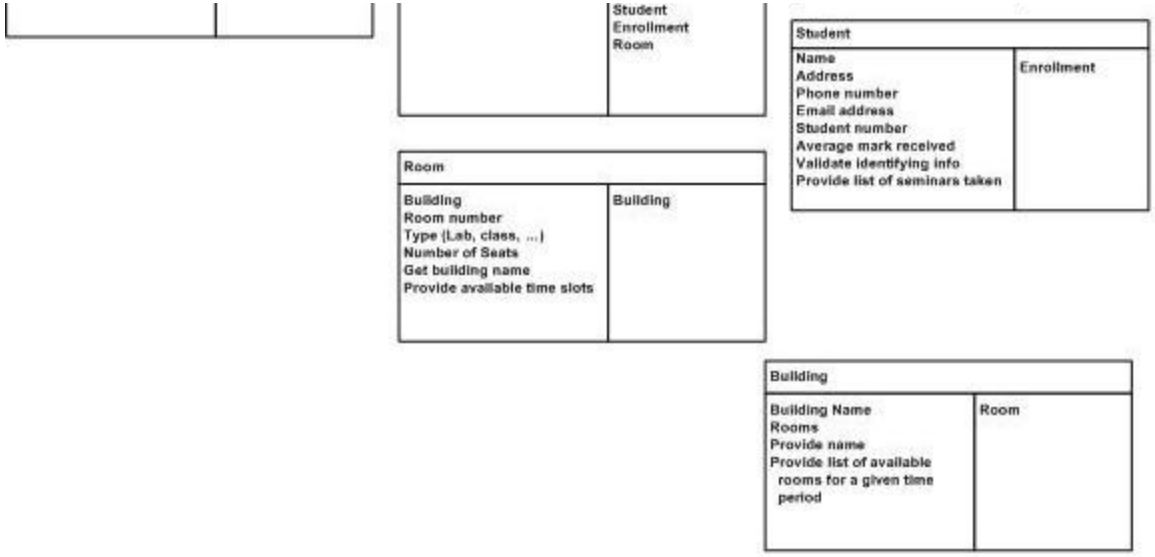
- **Find classes**. Finding classes is fundamentally an analysis task because it deals with identifying the building blocks for your application. A good rule of thumb is that you should look for the three-to-five main classes right away, such as *Student*, *Seminar*, and *Professor* in Figure 4. I will sometimes include UI classes such as *Transcript* and *Student Schedule*, both are reports, although others will stick to just entity classes. Also, I'll sometimes include cards representing actors when my stakeholders are struggling with the concept of a student in the real world (the actor) versus the student in the system (the entity).
- **Find responsibilities**. You should ask yourself what a class does as well as what information you wish to maintain about it. You will often identify a responsibility for a class to fulfill a collaboration with another class.
- **Define collaborators**. A class often does not have sufficient information to fulfill its responsibilities. Therefore, it must collaborate (work) with other classes to get the job done. Collaboration will be in one of two forms: a request for information or a request to perform a task. To identify the collaborators of a

class for each responsibility ask yourself "does the class have the ability to fulfill this responsibility?". If not then look for a class that either has the ability to fulfill the missing functionality or the class which should fulfill it. In doing so you'll often discover the need for new responsibilities in other classes and maybe even the need for a new class or two.

- **Move the cards around**. To improve everyone's understanding of the system, the cards should be placed on the table in an intelligent manner. Two cards that collaborate with one another should be placed close together on the table, whereas two cards that don't collaborate should be placed far apart. Furthermore, the more two cards collaborate, the closer they should be on the desk. By having cards that collaborate with one another close together, it's easier to understand the relationships between classes.

Figure 4. CRC Model.





How do you keep your CRC modeling efforts agile? By following the AM practice Model in Small Increments. The best way to do this is to create a CRC model for a single requirement, such as a user story, business rule, or system use case, instead of the entire collection of requirements for your system. Because CRC cards are very simple tools they are inclusive, enabling you to follow AM's Active Stakeholder Participation practice.

It's important to recognize that a CRC model isn't carved in stone. When you evolve it into a UML class diagram, or perhaps straight into code, you'll change the schema over time. Responsibilities will be reorganized, new classes will be introduced, existing classes will disappear, and so on. This is what happens when you take an

evolutionary approach to development.

Source

This artifact description is excerpted from Chapter 8 of The Object Primer 3rd Edition: Agile Model Driven Development with UML 2.