0:14

Note cards are often used when planning a speech. You can represent each of your talking points on a card. The speech only makes sense if you ordered the cards in a way that you move logically from one talking point to the next. It would be nice if we had something similar to map out the structure of the software logically when forming its design. The good news is we do. You identify components, connections and responsibilities from some requirements when forming the conceptual design. This is where you give your initial thoughts on how you might satisfy the requirements. In the technical design, you learned how these components and connections are further refined to give them technical details. This makes them easier to implement. Although identifying components, their responsibilities and connections, is a good first step in software design, we haven't yet demonstrated a way of representing them. Being able to play with our design, like reordering note cards and speech planning would be valuable. In this lesson, you will learn about an important technique for representing this information at a high level when forming the conceptual design. This technique uses CRC cards where CRC stands for Class, Responsibility, Collaborator. Similar to how note cards help you organize your talking points, CRC cards help you to organize your components into classes, identify the responsibilities and determine how they will collaborate with each other. To learn about CRC cards in a software design, let's have a mini software system we can play with. Consider a bank machine, for instance. You insert your bank card into the bank machine, the bank machine will then ask you to enter a PIN authenticating you for access. After that, you can choose to deposit, withdraw or check your balances. This scenario suggests the basic requirements for the system. Admittedly, it is an incomplete set of requirements but it's a good start. Remember that requirements are often incomplete and are resolved with further interactions with your client and end users. The next step is to design the bank machine. But as we form the conceptual design beyond just identifying components, their responsibilities and connections, we are going to represent them with our new technique, the CRC card. Much like how note cards are used to organize your talking points, CRC cards are used to record, organize and refine the components in your design. Again, CRC stands for Class, Responsibility and Collaborator. A CRC card has three sections. The top of the card has the class name. On the left are the responsibilities of the class, and on the right, you list collaborators. Collaborators are other classes that the class interacts with to fulfill its responsibilities. So, how can we use this while forming the conceptual design? To keep track of each candidate component and its responsibilities using a CRC card, you place a component's name in the class name section and the responsibilities in the responsibilities section. That's pretty straightforward so far. So, what about the connections? In the collaborators section, you list other components that your current component connects to or interacts with to fulfill its responsibilities. A CRC card can be as simple as a physical index card marked into three sections. They are cheap, editable and widely available. CRC cards are small on purpose, so you can't write much on them. This forces you to keep breaking down each component into smaller components and eventually, classes that are small enough to be individually described on index cards. Now that you've learned about CRC cards, let's use them to design our bank machine system. Let us begin with a basic user component. In this example, our primary user would be a bank customer that would go on our first CRC card. We place bank customer in the class name section. The bank customer's responsibilities are insert a bank card or choose an operation, such as deposit, withdraw or check account balance. Let's list these in the responsibility section of the CRC card. I'd go with insert bank card and choose operation, but write a description that makes sense for you. All of these bank customer responsibilities involve a bank machine. The customer can insert bank card in the machine and choose an operation. Since the bank machine is required for our bank customer component to fulfill its task, we placed bank machine under the collaborators section of the bank customer card. Next, let's do the other component, the bank machine on another CRC card. We write bank machine in the class name section. The machine's responsibilities include authenticate bank customer, display task options, deposit, withdraw and check balances. And since this card interacts with the bank customer component, add bank customer to the collaborators section of the bank machine card. As we have said, a key advantage of using CRC cards is that it allows you to physically reorganize your design. You can move related cards together or situate cards to suggest relationships. With our CRC cards, we can organize things by placing collaborating components together. For example, put the bank customer CRC card on the left and the bank machine CRC card on the right. Where CRC cards organized, you can simulate a prototype design of the system so far. Now, let's consider the scenario of the bank machine authenticating our bank customer. You can imagine enacting our bank customer's insert card responsibility. This in turn, collaborates with the bank machine and triggers its responsibility to authenticate the bank customer. Once authenticated, the bank machine enacts its responsibility to display the task options. Here is where CRC cards shine. They are cheap, editable and disposable, so you are encouraged to experiment and play with alternative designs. In the simulation we just did, you might ask, how does a bank machine authenticate the bank customer? This question suggests adding another component, the bank, where the bank machine communicates to authenticate the bank customer. Now, you should understand how working with CRC cards can help to identify needed components in the design. You can find more candidate components by using CRC cards to prototype and simulate various scenarios. For instance, since the bank machine will talk over a network to the bank, you can add a network CRC card between them. You want the network communication to be secure, so you make the network collaborate with a new component called encryption. This component supports secure communication with the bank. As well, you probably notice the bank

machine itself contains several different components, which seem small enough to be individual classes for programming. For example, there's a card reader, keypad, display, cheque slot and cash dispenser. Each of these classes but their responsibilities and collaborators can be described on their own cards. In a design meeting with the software development team, you can have all the cards on the table and discuss a simulation of how these classes work with other classes to achieve their responsibilities. As before, these simulations may reveal shortcomings in the design and you can experiment with alternatives by introducing appropriate cards. In this lesson, you've learned how to use CRC cards in designing software. You've also learned how CRC cards can be used for prototyping and simulation. This allows you to reveal shortcomings in the requirements or design. Again, CRC cards are just a technique for representing candidate components, their responsibilities and connections. Using note cards won't guarantee a great speech and you still need to be experienced in speech writing to be able to arrange your talking points properly. Similarly, you still need to be knowledgeable of various design techniques in order to define your CRC cards and form your design properly. Throughout the rest of this specialization, we will discuss various techniques so you can design more effectively.