System Sequence Diagram

The development of an SSD based on an activity diagram falls into four steps:

1. **Identify the input messages**. If in the corresponding Activity Diagram, there are three locations with a workflow arrow crossing the boundary line between the actor and the system, there needs to be three input messages in the System Sequence Diagram. At each location that the workflow crosses the automation boundary, input data are required; therefore, a message is needed.

2. **Describe the message from the external actor to the system by using the verb-noun syntax and using an appropriate message notation that makes the purpose clear**. In most cases, you will need a message name that describes the service requested from the system and the input parameters being passed. Assuming an SSD for a *Create customer account* use case where three automation boundary workflow crosses—Request account, Enter address, and Enter credit card info—were in its activity diagram, the SSD must illustrate the same three messages. Ensure that the names of the messages reflect the services that the actor is requesting of the system: createNewCustomer, enterAddress, and enterCreditCard, for example. Other names could also be used; instead of enterAddress, the name could be createAddress. The point to remember is that the message name should describe the service requested from the system and should be in verb-noun form.
   The other information required is the parameter list for each message. Determining exactly which data items must be passed in is more difficult. In fact, developers frequently find that determining the data parameters requires several iterations before a correct, complete list is obtained. The important principle for identifying data parameters is to base it on the class diagram. In other words, the appropriate attributes from the classes are listed as parameters. Looking at the attributes, along with an understanding of what the system needs to do, will help you find the right attributes. With the first message just mentioned—createNewCustomer—the parameters should include basic information about the customer, such as name, phone, and e-mail address. Note that when the system creates the customer, it assigns a new customerID and returns it with the other customer information.
   In the second message—enterAddress—parameters are needed to identify the full address. Usually, that would include street address, city, state, and zip code. The SSD simplifies the message to show "address" as the parameter. Unlike an ERD for database design, compound attributes are permissible.
   The third message—based on the activity diagram—enters the credit card information. The parameter—cc-info—represents the account number, expiration date, and security code.

3. **Identify and add any special conditions on the input messages, including iteration and true/false conditions**. In this instance, the enterAddress message is repeated for each address needed for the customer (e.g., delivery address, billing address). The asterisk symbol in front of the message is shown, indicating a looping of the message.

4. **Identify and add the output return messages.** Remember that there are two options for showing return information: as a return value on the message itself when using this option to represent a looping message as opposed to the loop frame, or as a separate return message with a dashed arrow. The activity diagram can provide some clues about return messages, but there is no standard rule that when a transition arrow in the workflow goes from the system to an external actor that an output always occurs. In the existing example activity diagram, there are three arrows from the System swimlane to the Customer swimlane. In the corresponding SSD example, these are shown as return data on the dashed line for two returns and as a returned value on the same line as the looping message for enterAddress. Note that they are each named with a noun that indicates what is being returned. Sometimes, no output data are returned.