

The Inside Track
Please Step This Way, Part 2
October 2001

As I write this, less than a week has passed since the events in New York, Washington DC and Pennsylvania. There should be an adjective before the word “events” in the prior sentence, but I am at a loss for the right one. Candidates include “tragic” and “disastrous” amongst others, none really seem to fully encompass how I feel about those events. Emotions range from anger and hatred to bottomless sadness and emptiness. I image we will all go through many more emotions before we find a way to deal with these events as a part of our lives.

Last month, I used the word “groovy” as a catchword to describe a process. Coincidentally, I think of that word’s heyday as being during the Vietnam War. I also think of the Vietnam War as our last “big” war. I’m not interested in debates about declared vs. undeclared, but the President has said that we are now at war. What I am interested in is a topic of much discussion: how will these events change the USA? One change I hope we can agree we don’t want to see is the loss of fundamental freedoms. It is highly probable that we will lose some privacy when we go to public locations like a stadium or airport, but we should not lose the right to personal safety, respect and human decency. To this end, I would encourage you to read and consider “signing” a pledge to “Stop the Hatred” available at:

<http://www.StopTheHatePledge.com>

Essentially the pledge asks that we, in the words of Martin Luther King, Jr., judge people not by the color of their skin, but by the content of their character. These recent events have been compared to Pearl Harbor. Do you suppose we can avoid making the same kind of mistake we made then, i.e. the internment camps for Japanese Americans?

The Next Step

Last month, this column concluded after step 2 of a 5 step software development process that uses UML as its analysis, design and communications vehicle. To review, step 1 was “define”. We defined the requirements of the system by using Use Case diagrams. In step 2, we refined each requirement by drawing activity diagrams. An activity diagram shows the steps needed to fulfill the use case. Knowing that, it won’t be a surprise that this month, we start with step 3!

Step 3 - Assign

Once the Activity Diagram is drawn in step 2, you're ready for step 3, assign. In the assignment step you identify objects or interfaces in your system to carry out each activity. You could assign an activity to a Processor (CPU) instead. This would be appropriate if you were designing a distributed system or a multithreaded application. Figure 1 is my Activity Diagram with interfaces assigned using "Swimlanes".

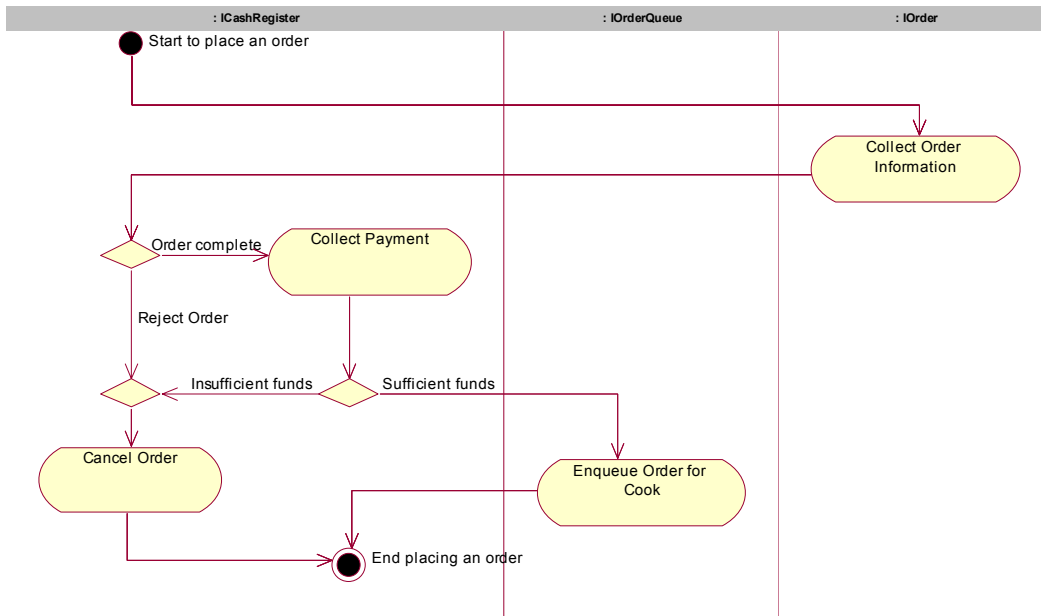


Figure 1 - Activity Diagram w/Swimlanes

Since I have no preconceived idea about what I'm building, I tentatively decided on 3 interfaces: ICashRegister, IOrder and IOrderQueue. Swimlanes are added to the Activity Diagram and each activity is drug into a particular swimlane. Through this process, you are assigning each activity to an object or interface that will implement it. As you can see in the diagram, a swimlanes looks the way it sounds, parallel vertical lines with labels at the top. Each activity lives in a swimlane, labeled with my interfaces.

Step 4 - Design

Step 4, design, finally a step with a familiar name! Using the objects and interfaces assigned in step 3, we can design components to implement them. Components will have dependencies between one another that you can identify based on the swimlanes in an activity diagram. Draw the dependency lines from each component to the interface it is dependent on. You can see in Figure 2 my component diagram is pretty simple with one interface assigned to each component.

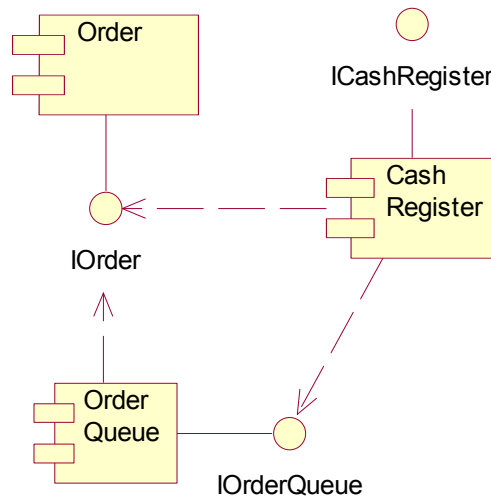


Figure 2 - Component Diagram

The boxes with the tabs are components and the “lollipops” are interfaces.

To each interface, add methods that implement the actions implied by your activity diagram. In Figure 3 you will see the methods I've added to my interfaces. I did a little thinking ahead. My activity diagram shows only "Enqueue Order for Cook" but my IOrderQueue interface contains a Dequeue method. As you added more use cases and assign the actions in an activity diagram to an interface via a swimlane, this is what you would expect. Your interfaces will build naturally. What more could you want from a process? Groovy!

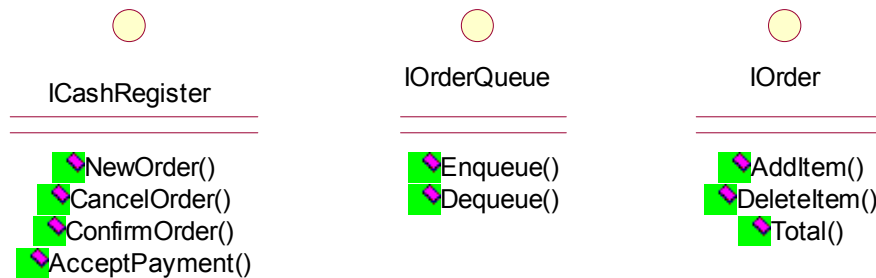


Figure 3 - Interfaces w/methods

Step 5 - Repeat

Now that we've completed 4 steps, we're ready to do it all over again. You are ready aren't you? Since we're going to repeat, we're going for another level of detail. Kind of like a recursive algorithm, it's a recursive process! As you well know, unless you're into endless loops, every recursive algorithm must have a well-defined, reachable end. I wish there was a number or some other magic formula for ending this process. You will have to think, you will have to decide, because only you understand the problem you are trying to solve. You are done when you have reached the level of detail in your UML model you feel is necessary to communicate your design to a developer. You are the terminator. But don't let it go to your head, Arnold Schwarzenegger you ain't.

Since step 1 of our process was Define, we'll repeat that. We'll create a Use Case Diagram for an actor; the actor will be an interface or object. The use cases in the diagram will correspond to the methods of the interface or object. I've included an example in Figure 4.

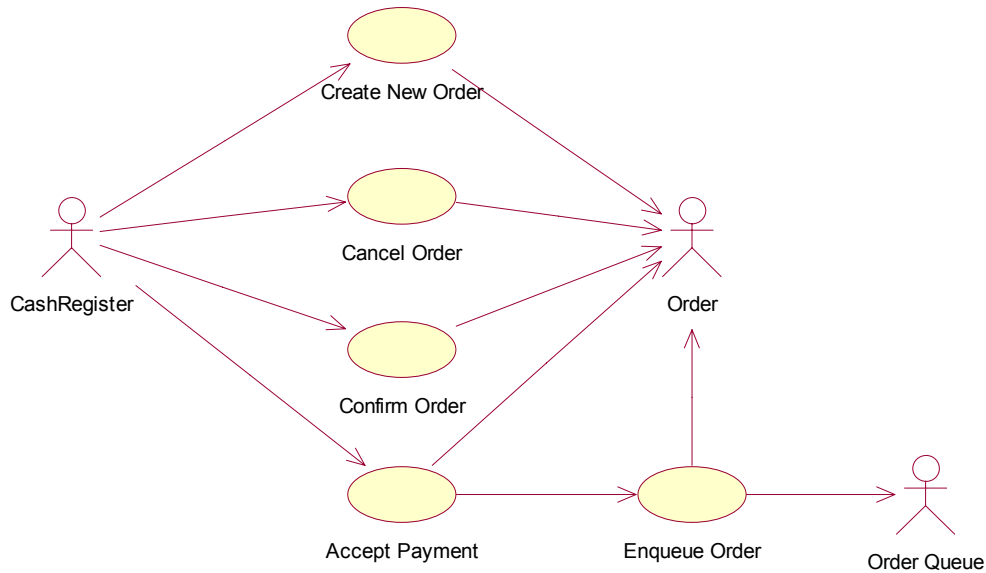


Figure 4 - Define Again

Next, taking each use case, you repeat step 2, Refine, by creating activity diagrams. In step 3, Assign, the activities are assigned to swimlanes. Depending on how deep you are in your design, each swimlane may now correspond to an actual class. Just like the interface on a component, a class will have methods which you Define in step 4. And back to step 5 again, the Repeat step. Just don't forget to stop!

The Final Step

With this edition of The Inside Track my run as the Microsoft Great Plains Platform Services Technical Evangelist (MSGPPSTE) comes to an end. You see, there's a rule that anyone who has a title based on an acronym of 7 letters or more can only serve in the role for one year. OK, maybe not. But you must have noticed that I said more than once that "I like code" and I'm moving back to a development role.

If you have been associated with Microsoft Great Plains for any period of time, you will also have noticed that learning is highly valued. The Inside Track may continue on, using another columnist or perhaps guest columnists, but whatever happens you can be guaranteed that a ton of information will be coming your way.

Look me up at the next Tech Conference and keep writing code!

Karl Gunderson
Former Technical Evangelist
solutiondeveloper@greatplains.com