

The Inside Track
Please Step This Way, Part 1
August 2001

As a professional developer for over 20 years now (yikes!) at times I think maybe I've fallen into a groove. No, no not a rut, a groove. I'm just a tad too young to have used groovy in my daily vocabulary as a young adult, but I certainly heard the term. Groovy is a positive term that can be applied to an emotional state (did the song "Feeling Groovy" just pop into your head?) or it can be used in much the same way as "cool" is today. The mellow term groovy is far more laid back than the harsher, more energetic cool. Someone, I've forgotten who, tried to explain the origin of the word groovy by an analogy to something fitting into a groove, evoking phrases like "works well", "fits like a glove" or "in the groove".

In The Groove

If you are in the groove, then you are likely to stay there. Being in the groove, one thing follows naturally from another. Sometimes when developing software, one thing seems to lead naturally to another, but not always.

At times, when I've been in the software groove, it's because I had a clear vision of what I wanted to accomplish and did it. The next step was "obvious" and natural. I wish I could bottle those feelings. I'd sell it as the "Groovy Software Development Oil". I'd make a mint. Well, maybe not.

Remember my March column? That was only 6 months ago! I wrote about UML and the 4 C's for developers. This month we're going to talk about UML again, as promised, but this time we're going to see some UML diagrams. UML is a language for communication as I mentioned but a language that uses diagrams as its vocabulary. Having a vocabulary isn't enough to communicate, however; you also need some sentence structure and language rules. You need a process!

The Process Groove

What is a process? It's the closest thing we have to "Groovy Software Development Oil". A process is a series of steps that leads from concept to finished software. Now there's no one process that's perfect for every situation, developer or organization. In my early days as a developer, I saw the manuals for SDM 70. I believe it stood for Software Development Methodology of the 70's and the binders took up a number of feet of shelf space. Process is a BIG subject. Some processes are huge. Microsoft Great Plains is working on a process that's a healthy size, but that's appropriate because we build pretty complex software.

If we're going to talk about UML as a tool for developing software, it really helps to have a process to give some structure to the diagrams that we draw. I've

drawn some UML diagrams, used them to design software. I've been introduced to the process we using at MSGP. But it wasn't until I attended the UML Boot Camp (www.umlbootcamp.com) taught by Martin L. Shoemaker that I understood why I wanted to draw a particular diagram at a particular time and what came next. A process gave the diagrams some "motivation" for existence. Process WAS the groove.

5 Step UML

In a week, Martin actually talked about 3 processes. The first he described as 5 Step UML. Martin said this process was so lightweight that it was only appropriate if you had no other process in place and were a team of no more than two. Since there are only the two of us and since we have never established a process that we can agree on, I think 5 Step UML is just about perfect. Perhaps we'll look at the full MSGP process in a future column, maybe.

Here is 5 Step UML:

1. Define
2. Refine
3. Assign
4. Design
5. Repeat

Martin's presentation of 5 Step UML used over 20 slides and included several demonstrations and exercises. I'll cover it lightly and hope that I've motivated you to learn more about UML. Martin's favorite book on UML is UML Distilled by Martin Fowler and Kendall Scott. If it's good enough for Martin, it's good enough for me.

Since I'm going to draw some actual UML diagrams, I'll need a system to design. I have in mind a software system for a fast food restaurant. Depending on how you think about such a problem, you could take one of several approaches.

- 1) I know little about the fast food business and want to study the interaction between customers and workers so that I can build a system appropriate for use in any fast food restaurant. I want to use standard PC components.
- 2) I know how customers and workers interact, I need to design software to support the interaction and then I'll build some hardware to implement the software system.
- 3) I have already decided on the hardware and want to design an order station and a cook station and network them, I need to design the software for this system.

Depending on familiarity with the problem and the design freedom available (Has the hardware already been selected? Do I have a budget and timeline?), you may start with 1, 2 or 3. Eventually you will most likely want to use UML to model

the software from an abstract point of view, a Logical Model in UML and how the software will run on the hardware, a Deployment Diagram in UML. Since my only experience with the fast food industry is as a consumer and I have no constraints, I'll start from point of view of the interactions between a customer and fast food worker, number one above.

Step 1 - Define

Many software development processes start with requirements and 5 Step does too. In the Define step, you identify requirements with Use Cases. A Use Case involves some external or Primary Actor that requests a service from the system under analysis. The Use Case involves providing the requested service perhaps relying on what is called a Secondary Actor to provide the service. Figure 1 is my first Use Case Diagram, there would be many more.

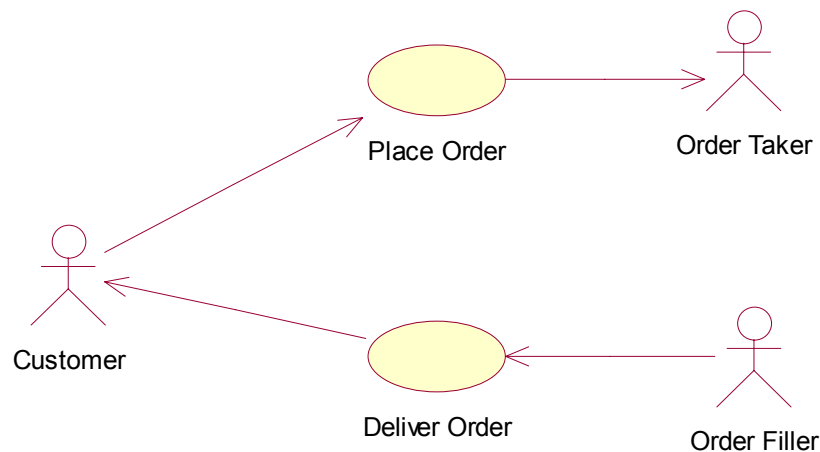


Figure 1 - Use Case Diagram

In my Use Case Diagram, the Customer is the Primary Actor that wants to order food, the Order Taker is a Secondary Actor and the use case is called Place Order. Actors often take the form of little stick people as show above, other icons are possible. A Use Case is represented by an oval with a caption. The arrows show the flow through the Use Case.

Step 2 - Refine

Now that we have defined a requirement that a customer must be able to place an order we can proceed to the next step of 5 Step UML, refine.

Remember flowcharts? I'm trying to forget them too. In the refine step the objective is to elaborate one of the Use Cases with an Activity Diagram. An

activity diagram looks a lot like a flow chart. For instance, the Activity Diagram for the Use Case “Place Order” might look like:

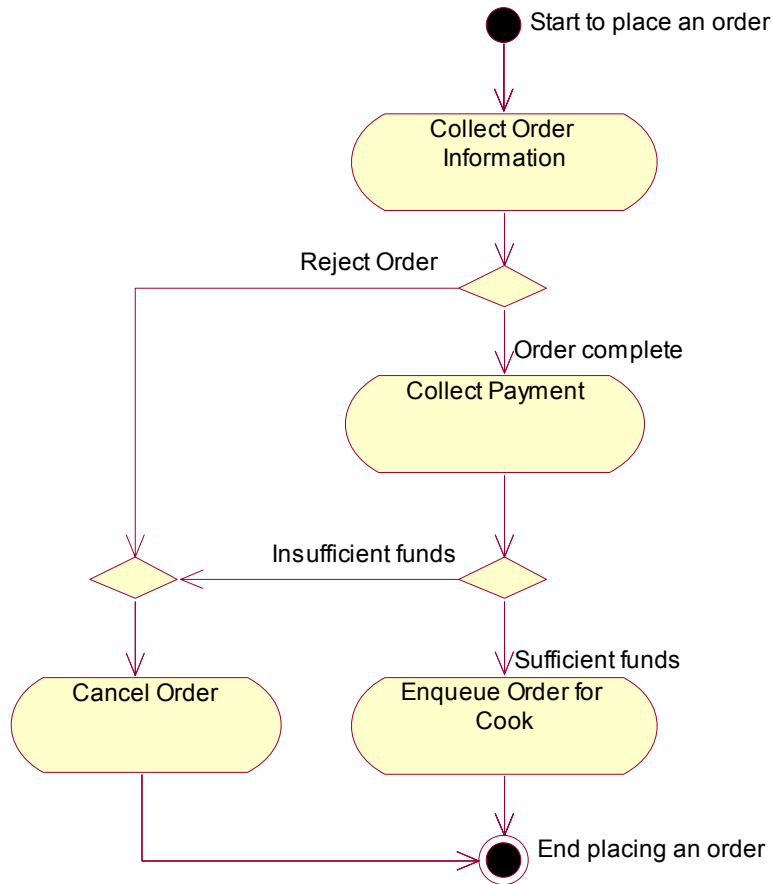


Figure 2 - Activity Diagram

The black bullet labeled “Start to place an order” is a Start State; there should be one per diagram. The “tubes” in the diagram are the activities. Each activity represents a step in completing some task. Often you will hear about scenarios in UML. A success or main scenario is the most common case where everything works. Alternate or failure scenarios represent all the other flows through a Use Case. These concepts are useful and are sometimes diagramed with a Use Case diagram. Another way of handling these scenarios that may work better for developers (it does for me!) is these Activity Diagrams. You can see in my activity diagram that the “main” scenario would be straight down from “Collect Order Information” to “Enqueue Order for Cook”. Alternate scenarios are handled by the diamond shaped branch and merge elements. The labels on the branches are similar to a decision in a flow chart, helping you determine which path to follow through the diagram. The black bullet in a ring is an End State.

You can have any number of End States, including none if you have a non-terminating or continuous activity. I only needed one which is labeled “End placing an order”.

Part 2 – To Be Continued

Hoping to leave you wanting more rather than feeling stuffed, we’ll finish our look at UML diagrams and the 5 Step UML process next time.

Till then, stay in the groove!

Karl Gunderson
Technical Evangelist
solutiondeveloper@greatplains.com