Using Abstraction and Encapsulation to make sandwiches Rule!

Encapsulation (*inˌkaps(y)əˈlāSH(ə)n*) : a group of related methods, properties, logic, and other members treated as a single object.

Abstraction (*abˈstrakSH(ə)n*) : the process by which everything other than the relevant data about an object is hidden in order to simplify use and increase maintainability.

There's a great <a>exercise used when teaching programming that demonstrates how important it is to create precise instructions. The exercise asks the participant to make explicit instructions about how to make a peanut butter and jelly sandwich, and then another person attempts to follow that plan to the letter - generally with highly entertaining results. This exercise can be a lot of fun to work through, whether you're at <a>Harvard or working with <a>kids, and it's a great way to learn how important it is to describe things well.

http://static.zerorobotics.mit.edu/docs/team-activities/ProgrammingPeanutButterAndJelly.pdf

https://www.youtube.com/watch?v=XWe4iohhmIw

https://www.youtube.com/watch?v=cDA3_5982h8

What it also demonstrates is the human tendency to unconsciously group things together and think of multiple steps as a single task - abstracting the details of execution so the caller doesn't need to worry about them. Rule Application creators can use that same human tendency to make the Applications easier to use and maintain.

So how do you make a sandwich?

Let's start at the most inclusive direction - Make a Sandwich. Whatever is requesting that sandwich really doesn't need to have any knowledge about the trivial details of how that happens - it just wants the delicious output of the process.

The average person's instructions for the construction of that sandwich may look something like this:

1. Gather bread, peanut butter, jelly, and a knife
2. Remove and lay out two slices of bread
3. Spread peanut butter on one slice
4. Spread jelly on the other slice
5. Put the two slices together
6. Eat the glorious result (optional)

Logically, everyone reading this article will understand these instructions and be able to interpret them to produce a delightful snack. However, each step listed is already an abstracted version of what that step requires. When building a Rule App, you need every step to be absolutely explicit and detailed.

No really, how do you make a sandwich?

Let's go one level of specificity down from there:

1. Gather bread, peanut butter, jelly, and a knife
    a. Bread must be baked
    b. Bread must be sliced into 1/2" thick slices cut perpendicular to the longest axis of the bread
    c. Peanut Butter must be in a jar with safety seal removed
    d. Jelly must be in a jar
    e. Knife should be a dull butter-style knife
2. Remove and lay out two slices of bread
    a. Bread container must be opened
    b. Bread should be places with the shortest axes perpendicular to the surface of the table
    c. Slices should be next to each other, in the same orientation, and not overlapping
3. Spread peanut butter on one slice
    a. Jar of peanut butter must be opened
    b. Requires indication of quantity to use
    c. Spreading must be on the upward-facing large sides of slice
    d. PB should be distributed in an even layer
    e. Knife should be used for this task, held by handle
4. Spread jelly on the other slice
    a. Jar of jelly must be opened
    b. Requires indication of quantity to use
    c. Spreading must be on one upward-facing large sides of slice without peanut butter
    d. Jelly should be distributed in an even layer
    e. Woah, you're actually reading through all these? Impressive.
    f. Knife should be used for this task, held by handle
5. Put the two slices together
    a. Jelly- and Peanut Butter-covered sides should face each other
    b. After combining, sandwich should be laid flat back down

Is it more specific? Heck yes. Is specific enough for a system with zero context to successfully execute? Nope!

Each one of those sub-items could be further clarified. And each of those... well, you get the point. Each step is an abstraction of the actual process needed - which makes it easier to understand for humans, but that's not enough for a Rule App.

Fine, let's pretend I made the most precise set of instructions ever. Now what?

At the end of all our elaboration of listing out each precise step, we end up with a giant list of incredibly precise steps of how to do everything (which is exactly what our application needs), and our instructions to make a sandwich include hundreds of steps. Will it get the right product every time? You bet! Will anyone in their right mind want to maintain it? Highly doubtful.

Why'd I go through all that elaborating then?

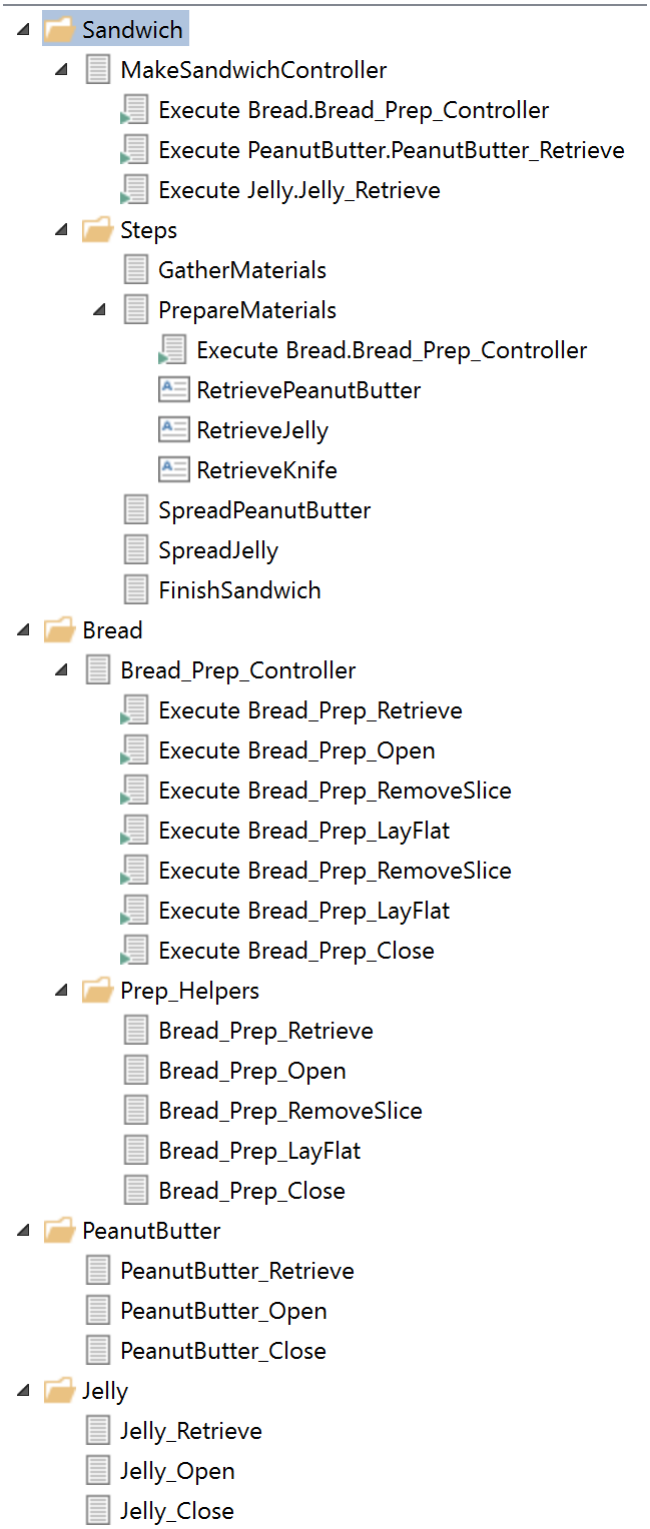
How do we get the level of precision required by rule applications while having something that's actually maintainable? By borrowing from human nature and abstracting the steps back into logical groups within the Rule Application! Folders, helper RuleSets, Entity Context and Vocabulary are all tools that we can use to shift this giant list of logic into groups that are easier to understand and read through for the humans that will actually be maintaining the application.

What was that you said about Entity Context?

One of the ways that you can group logic within a Rule App is by encapsulating the logic within its own entity context. Let the ingredients entity 'own' their respective logic of how to open the container. Let the knife Entity handle the intricacies of spreading material. Moving the logic into groups (Explicit RuleSets or Folders of RuleSets) and placing those groups into the context of item they interact with (Entities) allows the higher-order processes responsible for making the sandwich (Controller and helper parent RuleSets) to just tell the Peanut Butter to open, and the knife to spread the PB on that bread. Seems a whole lot easier to understand and maintain that sticking everything at the top level, doesn't it?

By grouping the details of logical steps into groups and Explicit RuleSets and encapsulating them into the context of the Entities they manipulate, we create a rule application that is drastically more understandable and maintainable. If you can't encapsulate the logic into child Entities, even using folders to organize RuleSets within the parent can help simplify management of a complex Rule App.

So go forth, and be not afraid to Encapsulate! You - and especially you in 3 years - will thank you.


Sandwich
MakeSandwichController
Execute Bread.Bread_Prep_Controller
Execute PeanutButter.PeanutButter_Retrieve
Execute Jelly.Jelly_Retrieve
Steps
GatherMaterials
PrepareMaterials
Execute Bread.Bread_Prep_Controller
RetrievePeanutButter
RetrieveJelly
RetrieveKnife
SpreadPeanutButter
SpreadJelly
FinishSandwich
Bread
Bread_Prep_Controller
Execute Bread_Prep_Retrieve
Execute Bread_Prep_Open
Execute Bread_Prep_RemoveSlice
Execute Bread_Prep_LayFlat
Execute Bread_Prep_RemoveSlice
Execute Bread_Prep_LayFlat
Execute Bread_Prep_Close
Prep_Helpers
Bread_Prep_Retrieve
Bread_Prep_Open
Bread_Prep_RemoveSlice
Bread_Prep_LayFlat
Bread_Prep_Close
PeanutButter
PeanutButter_Retrieve
PeanutButter_Open
PeanutButter_Close
Jelly
Jelly_Retrieve
Jelly_Open
Jelly_Close