**Workflow Foundation Internals (II)**

**Andrew Au**

Continue with our last article, we will work on separating program and data. Program and data are very distinct concepts from the perspective of a programmer. For example, you can run code, but you can’t run data. Code are not expected to be changed during runtime (except some circumstances such as hook or overlay), but data are expected to be changed all the time. From the processor’s perspective, they are no different. They are just some bytes. From an operating system perspective, the fact that code is read only allow optimization of main memory usage, because a single copy of the code can be used to run multiple processes.

Exactly the same argument applies for workflow. We think of Sequence as the program, but then sequenceCounter is just execution data. The fact that sequenceCounter is defined on the Sequence class made them inseparable. Consider multiple workflows are running on the same definition, the sequenceCounter will mix up and cause problem.

Reading the code for Activity, the fact that States is defined on the workflow is an even bigger problem. Instead of having them accessible from the activity as a protected field, we will retrieve it from the parameter of the Execute method. Similarly, we should not allow NextDelegate to be defined on the activity as well. This one is getting more complicated, as Sequence execution will require storage of two delegate objects. To solve this problem, we will use a Stack<Frame> holder for these objects. Diligent readers are recommended to perform the exercise himself, because we are going to take a quantum leap (instead of small refactorings) to move to the next version as shown below



*Code Sample: Version 6*

I tried to keep modification of Version 6 from Version 5 as small as possible. As we are moving the delegate storage from the activity to state, we also optimized Sequence execution. For now, executing a particular activity will not pass through sequence, but rather just saving the continuation of the sequence in the stack. Given the code is good now, I will refactor. In particular, we will move more logic into States so that the stack is maintained by state instead of being distributed in Sequence and Program. Making Frames a private field drives all these refactorings. All I did is really just giving the right name to the set of operations I did to the states.

*Version 7 has no structural change, just encapsulation of the stack.*

*Code Sample: Version 7*

We are getting very close now. For we are now serializing the state object, we should have completed the separation? The answer is NO. This is because the states contain a reference to the delegates, and then the delegates will link back to the program. To break this link, we cannot serialize the delegate anymore. We will serialize the MethodInfo and the activity ID as well, where activity ID can be obtained from a traversal of the activity tree. We will skip this part as this complicates our sample. Another problem is that the program is not constrained to be read only. There are multiple ways of solving it. WF3 chooses to make a copy of the program and always run the copy, while WF4 chooses to make a copy of the program (parts that is important to the runtime) and verify the main program is not modified at runtime. We could have also done by making user fail when program is changed at runtime as well. Again, this complicates the sample and we will stop our discussion here. Trying to implement or reading the existing implementation is a good way to appreciate this.

Another big problem with program and data is that a program line can be executed multiple times in a process. The simplest example is a loop. Those data has to be stored on the frame instead of the state. In fact, all states should be stored on a frame. Storing on States itself make them global variables. With frames, we can control variable scoping. This is one big improvement over the last version of WF.

Look at the implementation of States. One problem is ScheduleActivity can only be called once, because calling a subsequent call to ScheduleActivity will overwrite the NextDelegate field. In this next post, I will talk improve ScheduleActivity to allow multiple outstanding activities, and workflow-host communication through Bookmarks.