2011

University of North Carolina Wilmington

Master of Science in

Computer Science and Information Systems

Proceedings

https://csbapp.uncw.edu/mscsis

Implementing a Workflow Management System – Using Aspects

Edward J. Peloquin

A Capstone Project Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

Department of Computer Science / Department of Information Systems and Operations Management

University of North Carolina Wilmington

2011

Approved by

Advisory Committee

Dr. Ron Vetter

Dr. Douglas Kline

Dr. Devon Simmonds, Chair

Dean, Graduate School

Table of Contents

Abstract

Proper management of the business processes of organizations is central to profitability and survivability. Each business processes executes a sequence of activities and generates output to meet one or more organizational goals. While mapping and executing business processes in software has seen a dramatic increase in demand in recent years, mapping these processes to an execution runtime is a very complicated task. A system that handles this interaction must be able to translate the design time depiction of a process, specify what each business entity does, who is responsible, and validate the results of each process. In many cases, the software must also undergo changes and revisions much like the business processes they handle. When using a standard object oriented approach, these changes can result in a system where crosscutting concerns are more difficult to manage or maintain. Implementing a Workflow Management System using aspect oriented software development techniques, may allow the system to be better equipped to handle crosscutting concerns during the software development lifecycle but retain performance that is on par with a standard object oriented implementation.This project seeks to compare aspect-oriented and object-oriented approaches to the implementation of a workflow management system. To accomplish this goal the project will: (1) develop an object-oriented workflow management system based on the XPDL standard, (2) create an aspect-oriented version of the system by enhancing the system with aspects, and (3) compare the two systems based on execution time, memory usage, and the addition of a crosscutting concern.

1. Introduction

Ever since ancient times, people have been drawing pictures to help explain things [2]. The fact is that people understand pictures more than words [15]. By using illustrations; a complex process can be broken down into smaller, more manageable pieces therefore making it easier to understand, analyze, and improve [16]. This same premise is used today in business process mapping to help businesses improve their processes. It has been shown that process mapping techniques have resulted in billions of dollars of savings across multiple industries [2].

Many products currently exist to assist a business analyst in mapping a process. Some of these products include: Microsoft Visio [3], SmartDraw[4] and ConceptDraw [5]. Of course, modeling the process is only half the battle. In modern software based implementations, a graphical representation of a process is not enough to provide the level of automation necessary to actually facilitate the human and system interaction with a business process. For this to be accomplished, a process diagram must be linked with an execution runtime known as a Workflow Management System [2]. A Workflow Management System is as system responsible for automating the various stages of a process as represented by the processdefinition. Once the definition is imported, the workflow management system creates tasks, provides access to data, and automates other aspects of the business process in accordance with the imported schema. Like any other type of software, the efficiency and ease ofimplementingcrosscutting concernsare directly related to the techniques used to implement and manage them. I am taking Concerns to mean a particular set of behaviors needed by a program to function properly and Crosscutting concerns being concerns that cannot be cleanly decomposed from the rest of the system in both the design and implementation.

1.1 Problem Statement

Workflow Management Systems allow organizations to not only define and control the various activities associated with a business process but also provide the ability to measure and analyze the execution of the process [7]. Given the complex nature of these systems, design techniques can have a significant impact on their overall efficiency and the ease in which crosscutting concerns are added and maintained. By using standard object oriented approaches, the crosscutting concerns which inherently exist can become harder to understand and maintain since the logic to implement them isspread out over many classes. Since these systems play such a vital role in many businesses day to day operations, it is critical that they are made to withstand frequent updates in order to facilitate new functionality as required by the user community. Unfortunately, standard Object Oriented Programming techniques are known to have limitations when dealing with crosscutting concerns. One possible solution to this problem that I would like to explore is the use of Aspect Oriented Software Development techniques.

Aspect Oriented Software Development (AOSD) was designed to isolate and abstract crosscutting concerns into stand-alone modules called aspects [6]. These aspects (which closely resemble classes) help to encapsulate each concern in one place. In doing so, the number of lines of code in the application may be greatly reduced and the overall ease of maintaining the crosscutting concerns can be increased [6]. For example, instead of duplicating logging code in every method that needs it; an aspect can be developed and then applied such that the logging code is only written once.

1.2 Proposed Solution

In this project Ideveloped a Workflow Management System in whichlogging ishandled by aspects. In doing so I make comparisons on the overall ease in which this crosscutting concern is implemented and determine that the efficiency is on par with a standard object oriented approach. The process definitions for the system utilized the XML Process Definition Language (XPDL) format that has

been standardized by the Workflow Management Coalition (WfMC) [8]. For the process designer, I used TIBCO's Business Studio [9] to create the process models. I chose this software because it is free to use and it is capable of exporting the process models in XPDL format. The engine of the system was implemented with a Microsoft SQL 2008 database for storing the process data, .Net assemblies for the business logic, and aspects written with PostSharp [10] to encapsulate the logging concern.

Upon completion of the initial object oriented version system, I made a second system that made use of aspect oriented programming for the logging concern. Once developed, the two systems were tested to give quantifiable comparison metrics for efficiency and addition of the logging concern. The goal of this research wasto (1) Develop a workflow management system based on the XPDL standard, (2) Enhance the system with Aspects, and (3) Make a comparison of the two systems based on execution time, memory usage, and the addition of a crosscutting concern.

2. Background

The term workflow as described in ISO standard 12052:2006 *"consists of a sequence of connected steps. It is a depiction of a sequence of operations, declared as work of a person, a group of persons, and organization of staff, or one or more simple or complex mechanisms"* [11]. More commonly, workflow is known as a virtual representation of actual work being performed or in software development it is most often used to facilitate human-to-machine interaction. An example might consist of a document being routed around an office for approval. Each step in the approval process would represent one activity in the workflow.

The current notion of workflow as a concept has been around for a very long time. For many years, workflows have been used in many industries to increase throughput and resource utilization. Its origins can be traced back to H. Gantt and Frederick Taylor and their work on mass and energy flows as it related to the manufacturing industry. The most notable example of workflow as a concept is the assembly line [12] where a large process was broken down into several smaller discrete activities that, when completed in sequence, completed the larger entity. In today's business landscape, workflows are an essential way of life. They are used to achieve efficiency, reduce costs, and provide visibility to methods and processes such that improvements can be made.

2.1 Workflow Management Systems

Workflow Management Systems are software based systems that provide the necessary framework for process definitions (workflows) to execute. Most commonly these systems provide such functionality as (but are not limited to) [7]:

- **Process definition storage**: A mechanism for the storage and retrieval of a process definition.

- **Rules Based Decision Making**: The processing of business rules as defined by the workflow in order to determine how data is to be processed.
- **Document Routing**: The act of passing a file or a reference to a file from one participant to another.
- **Work lists**: A mechanism for a user to quickly identify their pending tasks.
- **Task automation**: Represents the computerized execution of a particular task. This may include email notifications or invoking an executable.
- **Event Notification**: Email or other electronic notifications are sent at pre-defined places in a process.
- **Process Monitoring**: The ability to monitor all aspects of a running workflow process.
- **Tracking and Logging of Activities**: Auditability functionality for monitoring workflow activities.

By providing the above listed features, a workflow management system can provide an organization with the necessary visibility of their processes to facilitate changes in order to reduce bottlenecks and increase organizational efficiency. More specifically, workflow management systems promote continuous business process improvement. Below is the Workflow Management Coalition's depiction of a full service workflow management system [13].

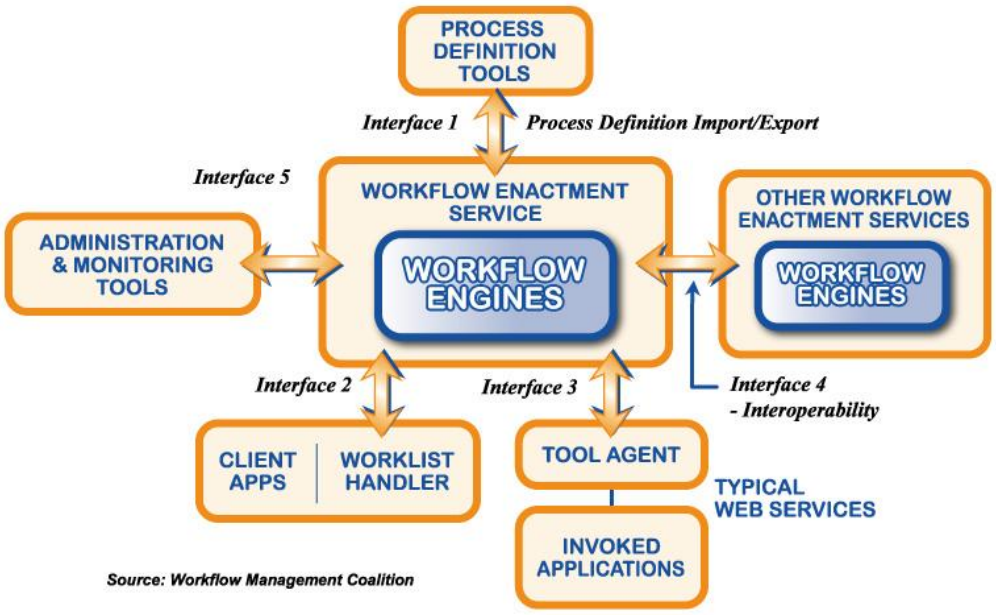

Figure 1: Workflow Management System Overview [13]

2.2 Software Quality Attributes

As defined in IEEE standard 1061, Software Quality is "the degree to which software possesses a desired combination of attributes." [17] As outlined in the IEEE standard, the main quality attributes consist of the following six attributes:

- **Efficiency:** The relationship of the level of performance to the amount to resources used under stated conditions.
- **Functionality:** The existence of certain properties and functions that satisfy stated or implied needs of users.
- **Maintainability:** The effort needed for specific modifications.
- **Portability:** The ability of software to be transferred from one environment to another.

- **Scalability:** The capability of software to maintain its level of performance under stated conditions for a stated period of time.
- **Usability:** The effort needed for use of the software.

In addition, each of these main attributes contains several sub-attributes. In many situations, designers must analyze and choose between many conflicting attributes to meet the user's requirements. This is known as "Software Quality Attribute Trade-offs" [18] and will vary greatly between systems. After careful consideration I've chosen to capture metrics specifically for Efficiency and a small subset of maintainability in which the addition of new functionality resides. I feel that these attributes are the most important for a workflow management system because this type of system needs to be efficient enough to handle large amounts of transactions as well as support frequent changes in functionality to satisfy customer demands. Testing Efficiency will determine the speed in which workflow activities can be executed and in the case of aspect oriented development it will help determine its feasibility as an alternative to standard object oriented approaches. Maintainability will determine how easy it is to make specific modifications to the system, in this case, the addition of a crosscutting concern. These are not the only important attributes to a system of this type but they are the most critical for success.

2.3 Aspect Oriented Software Development

Aspect Oriented Software Development was first introduced by Gregor Kiczales and his team at Palo Alto Research Center in the late 1990's [14]. As stated earlier, AOSD is aimed at isolating and abstracting cross cutting concerns into stand-alone modules called aspects [6]. In doing so, the number of lines of code in the application is greatly reduced and the overall maintainability is improved. AOSD allows developers the ability to change a cross cutting concern one place and to have it propagate throughout the system. It is also important to note that once an aspect has been defined and coded this

technology can and should be used with other similar applications. In order to achieve these benefits a number of new concepts were introduced with AOSD. Below is a brief description of the most important key terms:

- **Aspect**: Similar to classes in standard Object Oriented Software Development, an aspect is designed to encapsulate one concern in a centralized location.
- **Join Point:** A join point is a specification of when an aspect should be executed in the main program.
- **Pointcut:** A pointcut is a set of join points where a piece of code (called advice) associated with the pointcut is applied.
- **Advice**: Code associated with a pointcut that is then applied to all join points associated with the pointcut.

2.4 XPDL (XML Process Definition Language)

The XML Process Definition Language (XPDL) is a standardized format introduced by the Workflow Management Coalition (WfMC) in 2005. This standard was designed to provide a schema for specifying both the visual representation of a process and its semantic definition. Since 2005, the XPDL format has seen a number of revisions and is now currently in version 2.1 since 2008 [8]. Below is an image of an XPDL snippet, take notice of both the graphical and semantic data representations [8]. This format is very useful when transferring the process definition between systems that support the XPDL standard.

```
<ActivityId="10"Name="Transform Data">
<Implementation>
<Task>
<TaskApplicationId="transformData">
<ActualParameters>
<ActualParameter>orderString</ActualParameter>
<ActualParameter>orderInfo</ActualParameter>
</ActualParameters>
</TaskApplication>
</Task>
</Implementation>
<NodeGraphicsInfos>
<NodeGraphicsInfoPage="1"LaneId="0"Width="75.0"Height="50.0"BorderColor="-
16777216"FillColor="-1114150">
<CoordinatesXCoordinate="128.0"YCoordinate="96.0"/>
</NodeGraphicsInfo>
</NodeGraphicsInfos>
</Activity>
```

Figure 2: XPDL Snippet [21]

The Meta-Model for the XPDL schema is very broad. The schema encompasses a lot of functionality which makes it very flexible in its use. Below is the full meta-model of the XPDL Schema in figure 3. To keep the project size manageable I have chosen to only implement a portion of the full schema. The portion that I support is shown in figure 4.

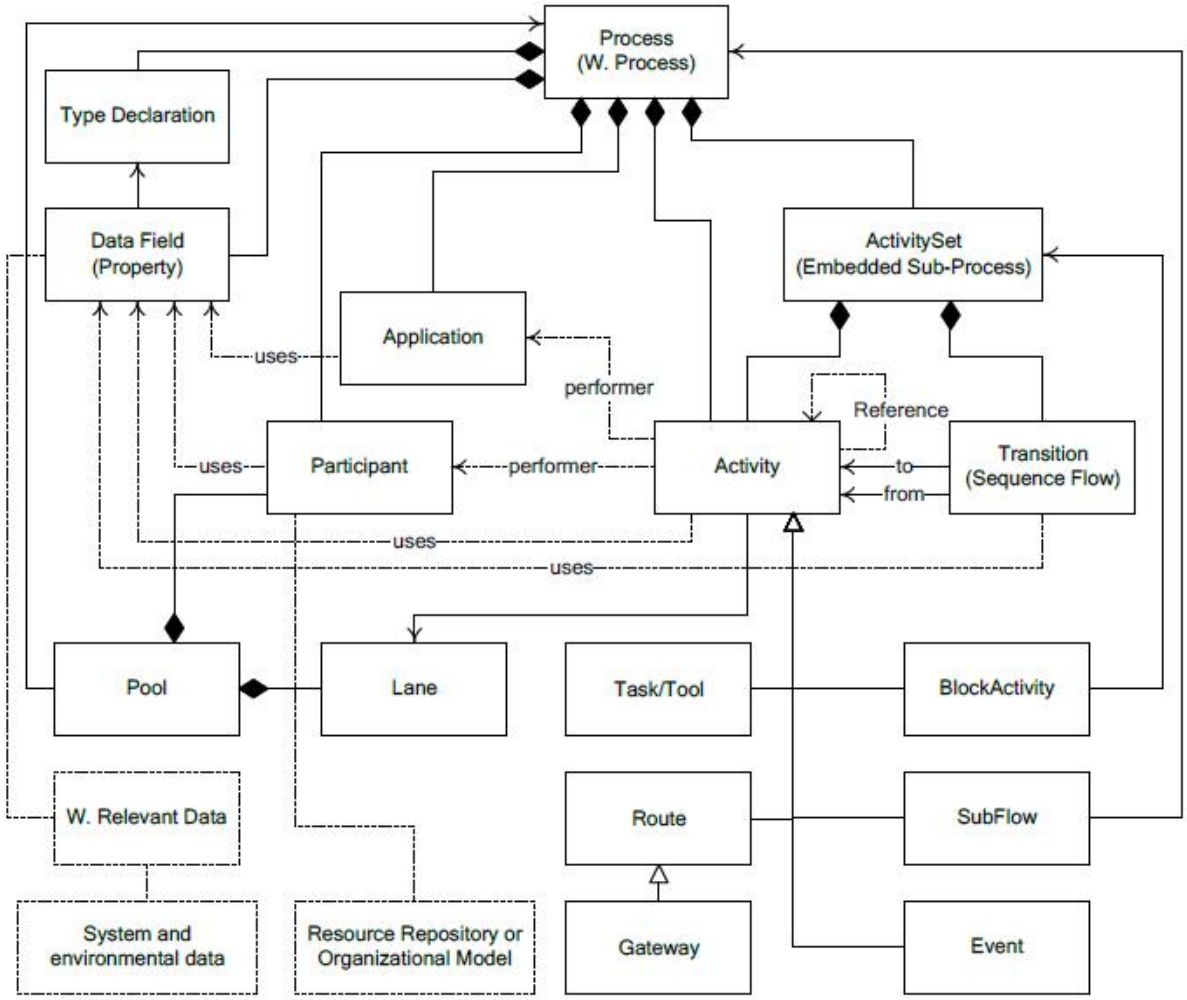

Figure 3: XPDL Process Meta-Model

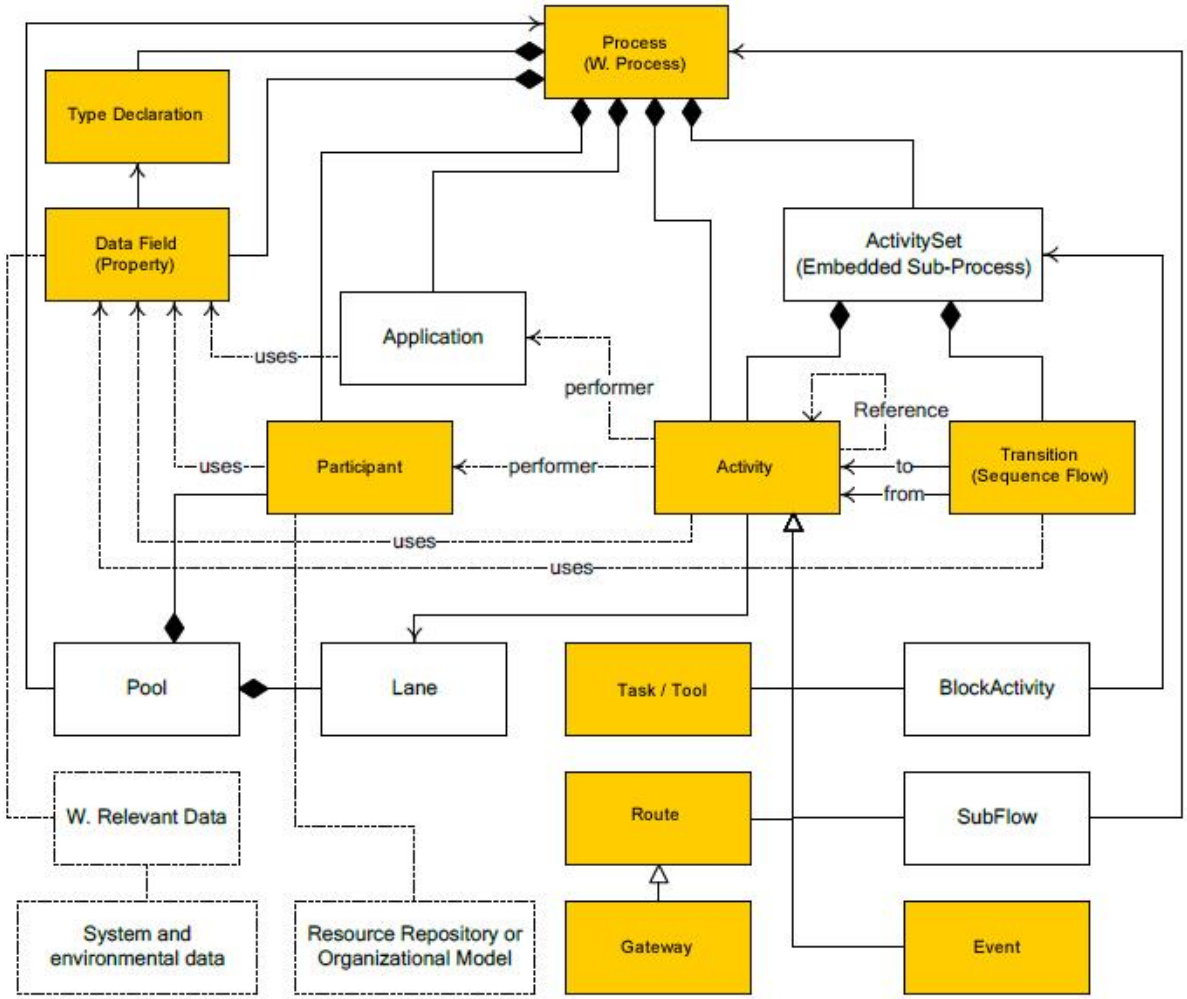

Figure 4: Supported Process Meta-Model

For this project I chose to focus on the main components of the XPDL Meta-Model (highlighted in Figure 4). In doing so I reduced the amount of complexity and time needed to implement the base system. In future revisions I would like to revisit this model and more fully implement it.

2.5 TIBCO Business Studio

TIBCO Business Studio is a free to use, eclipse based business modeling software. Designed with standards in mind, this software allows an analyst to visually design a business

process and then export it in a number of widely adopted standards such as the XPDL format which will be used for this project.

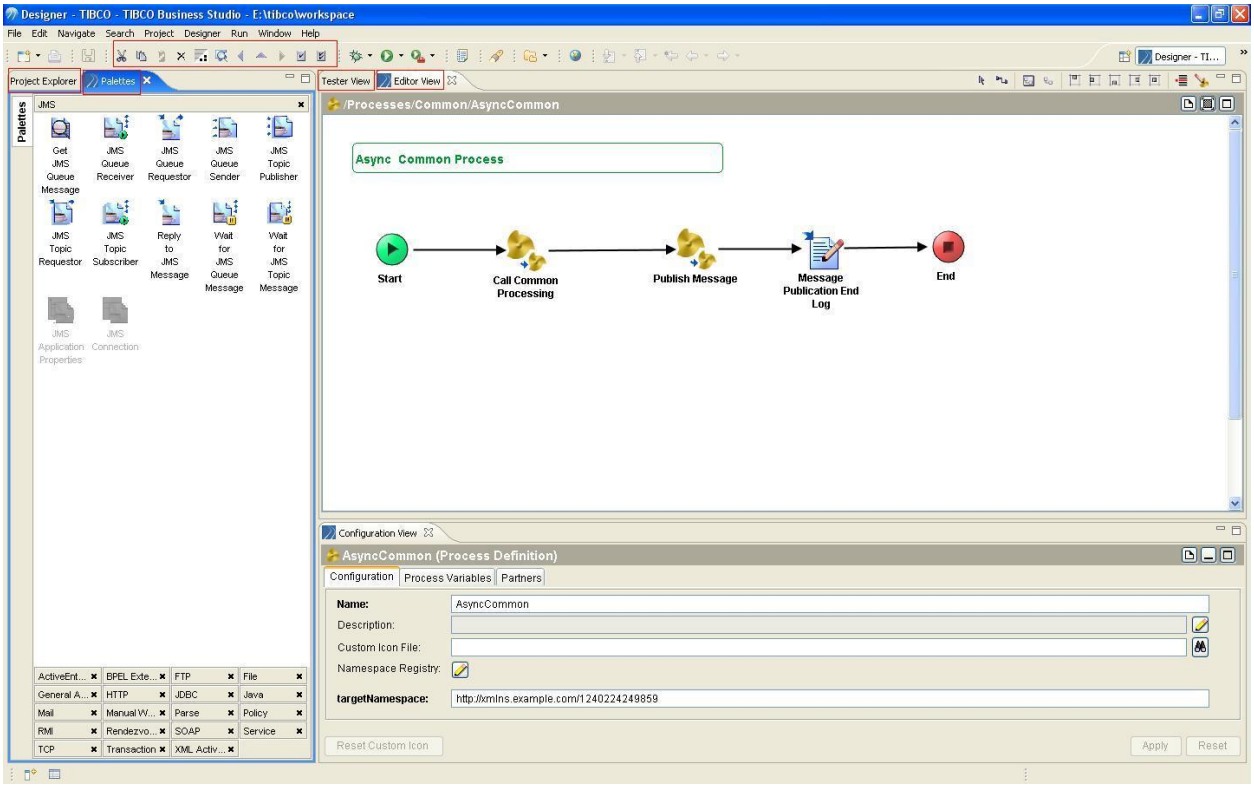

Figure 5: TIBO Business Studio

2.6 PostSharp

PostSharp is considered a “Post Compiler”. A Post Compiler modifies the target code after the main complier finishes and then injects the functionality that is needed (into the compiled dll). PostSharp was developed as a light weight (but not free), unobtrusive AOP framework. The main purpose of the framework is to help developers minimize the effort of implementing non-functional requirements such as transaction handling, tracing, caching, security, monitoring, and exception handling. Using this framework can help eliminate certain implementation patterns that are commonly repeated throughout applications. PostSharp contains seven base aspect types; below is a brief description of each [10]:

- **On Exception Aspect**: This aspect injects an exception handler inside the target method and allows for a standard way to implement exception handling policies.
- **On Method Boundary Aspect**: This aspect is extremely powerful. A method advised by this aspect is wrapped in a try / catch block and provides up to four additional event handlers (OnEntry, OnSuccess, OnException, and OnExit) allowing complete control over the target methods behavior. (see figure 6)

```
///<summary>
/// Aspect that, when applied on a method, emits a trace message before and
/// after the method execution.
///</summary>
[Serializable]
publicclass TraceAttribute : OnMethodBoundaryAspect
{
///<summary>
/// Method invoked before the execution of the method to which the current
/// aspect is applied.
///</summary>
///<param name="args">Information about the method being executed.</param>
publicoverridevoid OnEntry( MethodExecutionArgs args )
  {
    Trace.TraceInformation( "{0}.{1}: Enter",
                args.Method.DeclaringType.FullName, args.Method.Name );
    Trace.Indent();
  }

///<summary>
/// Method invoked after successfull execution of the method to which the current
/// aspect is applied.
///</summary>
///<param name="args">Information about the method being executed.</param>
publicoverridevoid OnSuccess( MethodExecutionArgs args )
  {
    Trace.Unindent();
    Trace.TraceInformation( "{0}.{1}: Success",
                args.Method.DeclaringType.FullName, args.Method.Name );
  }

///<summary>
/// Method invoked after failure of the method to which the current
/// aspect is applied.
///</summary>
///<param name="args">Information about the method being executed.</param>
```

```
publicoverridevoid OnException( MethodExecutionArgs args )
  {
    Trace.Unindent();
    Trace.TraceInformation( "{0}.{1}: Exception {2}",
                  args.Method.DeclaringType.FullName, args.Method.Name,
                  args.Exception.Message );
  }
}
```

Figure 6: PostSharp Code Snippet[11]

- **On Method Invocation Aspect:** This aspects behaves provides around advice for a target method. Calls to the target method are intercepted by this aspect and for the target method to execute the Proceed command must be invoked in the advice.
- **On Field Access Aspect:** This aspect intercepts all get and set operations on a target field. It provides two event handlers: OnGetValue and OnSetValue.
- **Composition Aspect:** This aspect facilitates the injection of new interfaces into a target type.
- **Implement Method Aspect**: This aspect allows the developer to defer the implementation of abstract or external methods.
- **Custom Attribute Injector:** This aspect adds a custom attribute to the target. For example, you could use this aspect to flag a method as Serializable.

2.7 ANTS Performance Profiler

The ANTS Performance Profiler [20] is a .Net code profiler that is used to analyze and improve the performance of applications. The profiler can be used by itself or in conjunction with Visual Studio. For my experiment I will use this tool to analyze method level performance of my application before and after applying the AOP techniques. I will be using this tool to calculate total execution time and memory usage for both versions of the system.

2.8 Microsoft Lines of Code Counter

Microsoft's Line of Code Counter [19] connects to Visual Studio Team Foundation, Visual Studio Services, or the file system and counts LOC based on configuration provided by the user. Some of the key features include:

- 16 types of count
- Customized counting
- Reports by file, folder, language, etc.
- Save counter projects for reuse
- Export results to Excel or PDF

For my experiment I will use this tool to analyze changes in LOC between the different versions of my application.

3. The Workflow System

From a high level, the workflow engine I ran my tests on is comprised of four main components: The main engine logic, process entities, data access classes, and shared utilities. All four components are written in C# and are contained in four different namespaces within the same assembly (dll). At runtime, a user application will communicate with the engine through the main engine class “WFEngine” via its public methods. Once invoked, the main engine class communicates with both the Process Entities and the Shared Utilities to traverse the workflow and execute its activities. After execution of each activity, the Process Entities communicate with the data access classes to persist the workflow data to the database (see figure 7).

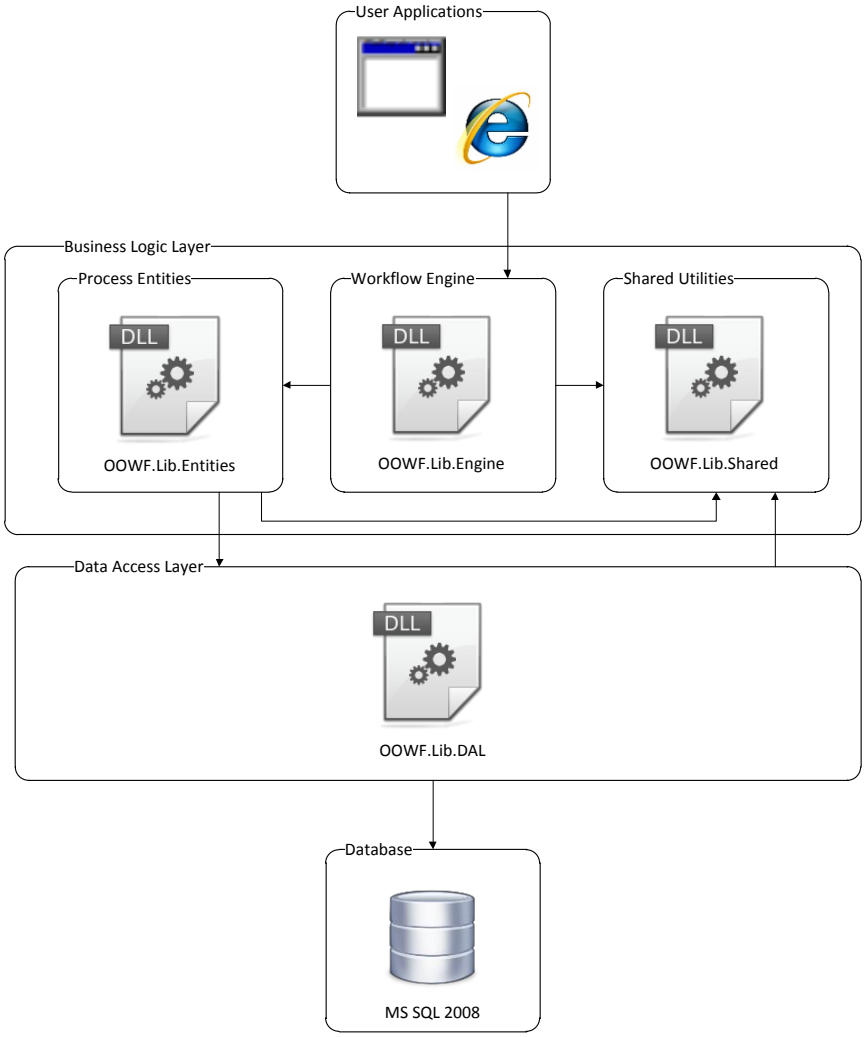

Figure 7: System Architecture Overview

3.1 Definition Entities

The Definition Entities in this system represent the workflows' object model based on the supported pieces of the XPDL Meta Model (figure 3). These Entities are all contained in the Lib.Entities.Definition namespace. All classes in the Lib.Entities.Definition namespace inherit from the BaseEntity class (section 3.1.1). The primary role of the Definition Entities is to be used as in memory lookups for the currently executing workflow. These classes contain no business logic. The actual implementation of the Entity is handled by the Runtime Entities (section 3.2). Sections 3.1.1 to 3.1.10

briefly describe each of the classes in the Lib.Entities.Definition namespace as shown in the class diagram in figure 8.

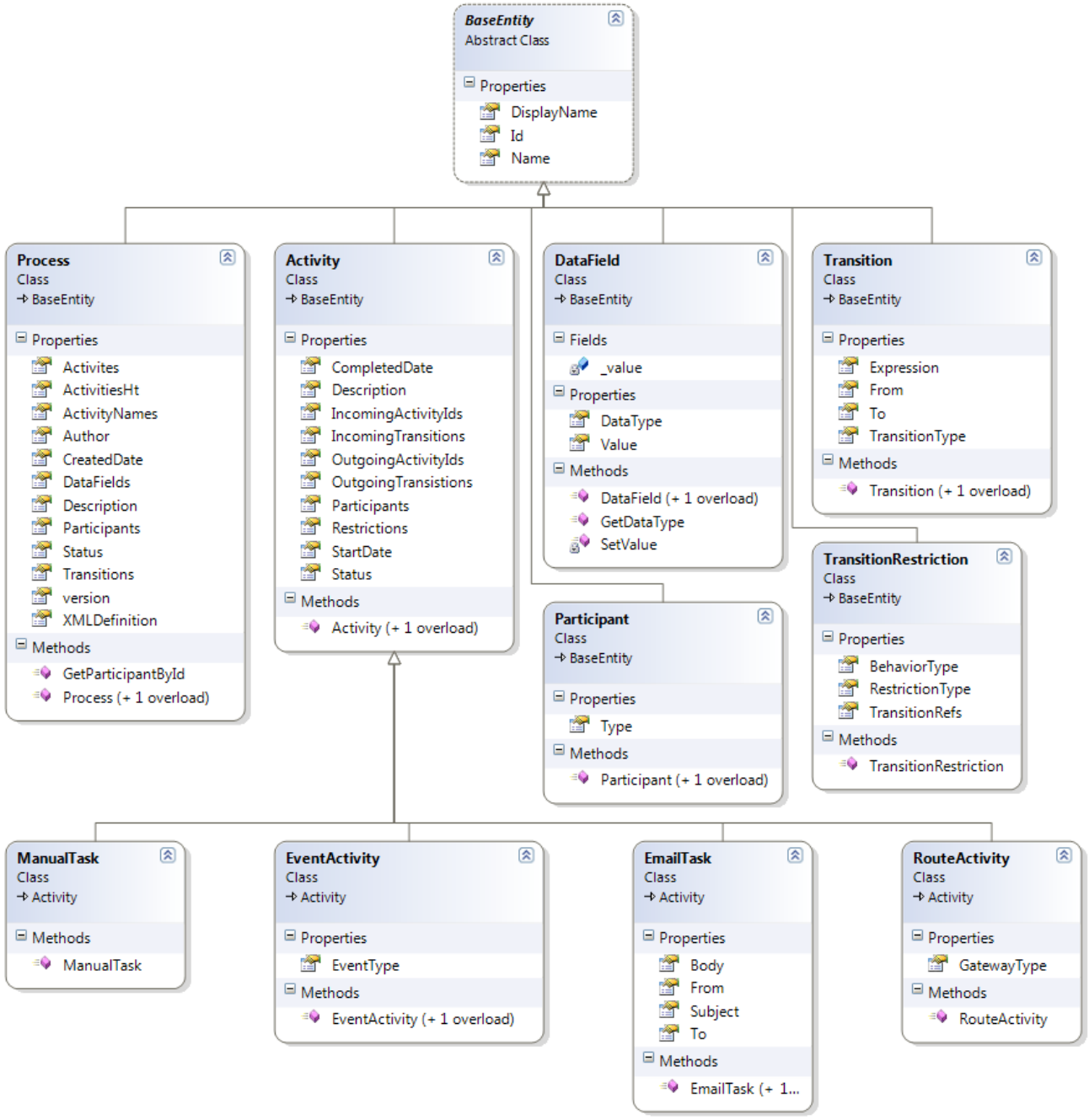

Figure 8: Definition Entities (Lib.Entities.Definition)

3.1.1 BaseEntity Class

The BaseEntity class is an abstract class in which all Definition Entities are derived from. The BaseEntity class contains public properties for id, name, and display name. The abstract RuntimeEntity class (Section 3.2.1) also extends from BaseEntity but adds more instance data and methods needed for runtime execution.

3.1.2 Process Class

The Process Class represents the entire definition of a workflow as outlined in the supported sections of the XPDL definition file. This class is primarily used to hold collections of workflow related objects such as Activities, Data Fields, and Participants. The main workflow engine class (section 3.3.1) references this class in order to traverse the workflows Activities at runtime.

3.1.3 Activity Class

The Activity class is an abstract base class in which all workflow activity definitions are derived from. The process class (section 3.1.2) holds a collection of all activities as defined by the workflows' definition. During runtime, the main engine class (section 3.3.1) resolves the abstract activity in the collection to its actual activity type. For example, the code below is used to determine the activity type and process it. This code was taken from the WFEngine class.

```
        ///<summary>
/// Determines the activity type and processes it
///</summary>
///<param name="a"></param>
privatevoid ProcessActivity(Activity a)
        {
if (a isEventActivity)
            {
this.ProcessEventActivity ((EventActivity) a);
            }
elseif (a isManualTask)
            {
this.ProcessManualTask ((ManualTask) a);
            }
elseif (a isEmailTask)
            {
this.ProcessEmailActivity ((EmailTask) a);
            }
elseif (a isRouteActivity)
            {
this.ProcessRouteActivity ((RouteActivity) a);
            }
else
            {
thrownewArgumentException("Unknown Activity Type: " +
                                                a.GetType().ToString());
            }
        }
```

Figure 9: Process Activity Method

Once the activity type is determined, a runtime version of the activity is created and executed. Below is one example of an XPDL Activity that this class is modeling. Details that determine what type of activity is defined reside in the child nodes of the activity tag. This example depicts an Event Activity.

```
<xpdl2:ActivityId="_UMtJ6VFtEeC7_uBwNhVvUA"Name="StartEvent"xpdExt:DisplayName="Start Event">
<xpdl2:Event>
<xpdl2:StartEventTrigger=”None”/>
</xpdl2:Event>
</xpdl2:Activity>
```

Figure 10: XPDL Activity Example

3.1.4 DataField Class

The DataField class represents a formal parameter in the workflows' definition. It contains properties to hold the data type and the value (which may be a lookup). This class is also used by the

ConditionEvaluator class (section 3.5.2) to evaluate gateway conditions. Below is one example of an XPDL DataField that this class is modeling.

```
<xpdl2:DataFieldId="_g7AGsFV5EeClTe_kgVEprw"xpdExt:DisplayName="Amount"Name="Amount">
<xpdl2:DataType>
<xpdl2:BasicTypeType="FLOAT">
<xpdl2:Precision>10</xpdl2:Precision>
<xpdl2:Scale>2</xpdl2:Scale>
</xpdl2:BasicType>
</xpdl2:DataType>
</xpdl2:DataField>
```

Figure 11: XPDL Data Field Example

3.1.5 Transition Class

The Transition Class is used to store links between activities in the workflow. An instance of the transition class stores information regarding what activity it is coming from and what activity it is going to. The transition object also stores the transition type which in an enumeration containing three values: None, Condition, or Otherwise. Below is the XPDL version of a transition that this class is modeling.

```
<xpdl2:TransitionId="_lWzeYFFtEeC7_uBwNhVvUA"xpdExt:DisplayName=""Name=""From="_UMtJ6VFtEeC7_uBwNhVvUA"To="_iMnaUFFtEeC7_uBwNhVvUA" />
```

Figure 12: XPDL Transition Example

3.1.6 Participant Class

The Participant class is used to store instances of the workflows participants. Instances of the Participant class can be one of two types: Human or System. Each Activity defined in the workflow contains a list of participants. Below is the XPDL version of a participant that this class is modeling.

```
<xpdl2:ParticipantId="_2fjvEFFtEeC7_uBwNhVvUA"xpdExt:DisplayName="Manager"Name="$ROLE{Manager}">
<xpdl2:ParticipantTypeType="HUMAN"/>
</xpdl2:Participant>
```

Figure 13: XPDL Participant Example

3.1.7 EventActivity Class

The EventActivity Class represents one type of Activity in the XPDL schema. Examples of Event Activities are the start and end event for a workflow. Figure 13 above shows an example of an Event Activity taken from XPDL.

3.1.8 ManualTask Class

The ManualTask Class represents one type of Activity in the XPDL schema. A manual task is assigned to a participant of type “Human”. This task type does not automatically complete until manually told to do so. Below is an example of a manual task as defined in XPDL.

```
<xpdl2:ActivityId="_iMnaUFFtEeC7_uBwNhVvUA"Name="ManagerApprovalTask"IsATransaction="false"
xpdExt:Visibility="Private"xpdExt:DisplayName="Manager Approval
Task"xpdExt:RequireNewTransaction="false">
<xpdl2:Implementation>
<xpdl2:Task>
<xpdl2:TaskManual>
<xpdl2:Performers>
<xpdl2:Performer>-defined-in-Activity-Performer-</xpdl2:Performer>
</xpdl2:Performers>
</xpdl2:TaskManual>
</xpdl2:Task>
</xpdl2:Implementation>
<xpdl2:Performers>
<xpdl2:Performer>_2fjvEFFtEeC7_uBwNhVvUA</xpdl2:Performer>
</xpdl2:Performers>
<xpdl2:Extensions/>
<xpdExt:AssociatedParameters>
<xpdExt:AssociatedParameterFormalParam="IsManagerApproved"Mode="INOUT"Mandatory="false">
<xpdl2:Description></xpdl2:Description>
</xpdExt:AssociatedParameter>
</xpdExt:AssociatedParameters>
<xpdExt:ActivityResourcePatterns>
<xpdExt:WorkItemPriorityInitialPriority="50"/>
</xpdExt:ActivityResourcePatterns>
</xpdl2:Activity>
```

Figure 14: XPDL Manual Task Example

3.1.9 RouteActivity Class

The RouteActivity Class represents one type of Activity in the XPDL schema. This activity type is used to represent conditional branches in the workflow. For this project I have chosen to only support XOR (Exclusive Or) gateways. XOR gateways are route activities in which only one outgoing path can be chosen. Below is an example of an XPDL Route Activity.

```
<xpdl2:ActivityId="__Q4awFFtEeC7_uBwNhVvUA"Name="Amount500"xpdExt:DisplayName="Amount >
$500">
<xpdl2:RouteGatewayType="Exclusive"MarkerVisible="true"ExclusiveType="Data"/>
<xpdl2:TransitionRestrictions>
<xpdl2:TransitionRestriction>
<xpdl2:SplitType="Exclusive"ExclusiveType="Data">
<xpdl2:TransitionRefs>
<xpdl2:TransitionRefId="_GZnYAFFuEeC7_uBwNhVvUA"/>
<xpdl2:TransitionRefId="_llUakFFuEeC7_uBwNhVvUA"/>
</xpdl2:TransitionRefs>
</xpdl2:Split>
</xpdl2:TransitionRestriction>
</xpdl2:TransitionRestrictions>
</xpdl2:Activity>
```

Figure 15: XPDL Route Activity Example

3.1.10 EmailTask Class

The EmailTask Class represents one type of Activity in the XPDL schema. This activity type is used to represent emails in the workflow. The EmailTask class contains all the data required for sending an email such as “To”, “From”, “Subject”, and “Body”. Below is an example of an XPDL Email Activity.

```
<xpdl2:ActivityId="_wtGAEFVxEeCITe_kgVEprw"Name="SeniorManagerRejectionEmail"IsATransaction="
false"xpdExt:Visibility="Private"xpdExt:DisplayName="Senior Manager Rejection
Email"xpdExt:RequireNewTransaction="false">
<xpdl2:Implementation>
<xpdl2:Task>
<xpdl2:TaskServicexpdExt:ImplementationType="E-Mail"Implementation="Other">
<xpdl2:MessageInId="_aLxycFV1EeCITe_kgVEprw"/>
<xpdl2:MessageOutId="_aLxycVV1EeCITe_kgVEprw"/>
<email:Email>
<email:Definition>
<email:Fromemail:Configuration="Server"/>
<email:To>${ProcessInitiator}</email:To>
<email:Subject>Your Expense Report has been rejected by your senior manager</email:Subject>
</email:Definition>
<email:Body>Your Expense Report has been rejected by your senior manager</email:Body>
<email:SMTPemail:Configuration="Server"/>
</email:Email>
</xpdl2:TaskService>
</xpdl2:Task>
</xpdl2:Implementation>
<xpdl2:Performers>
<xpdl2:Performer>_C20eMlV1EeCITe_kgVEprw</xpdl2:Performer>
</xpdl2:Performers>
<xpdl2:Extensions/>
<xpdExt:ActivityResourcePatterns>
<xpdExt:AllocationStrategyxpdExt:Strategy="SYSTEM_DETERMINED"/>
</xpdExt:ActivityResourcePatterns>
</xpdl2:Activity>
```

Figure 16: XPDL Email Task Example

3.2 Runtime Entities

The Classes that make up the set of runtime entities are responsible for modeling the systems runtime data such as process instances, activity instances, and versioned process definitions. Each runtime entity is modeled after a definition entity. Although the mapping is not exactly one to one, it is close. Each runtime entity contains a method call Execute() in which the actual implementation occurs. The runtime entities are also responsible for communicating with the classes in the data access layer to retrieve and save the runtime data in the database. Figure 17 shows the full class diagram for the classes that make up the runtime entities.

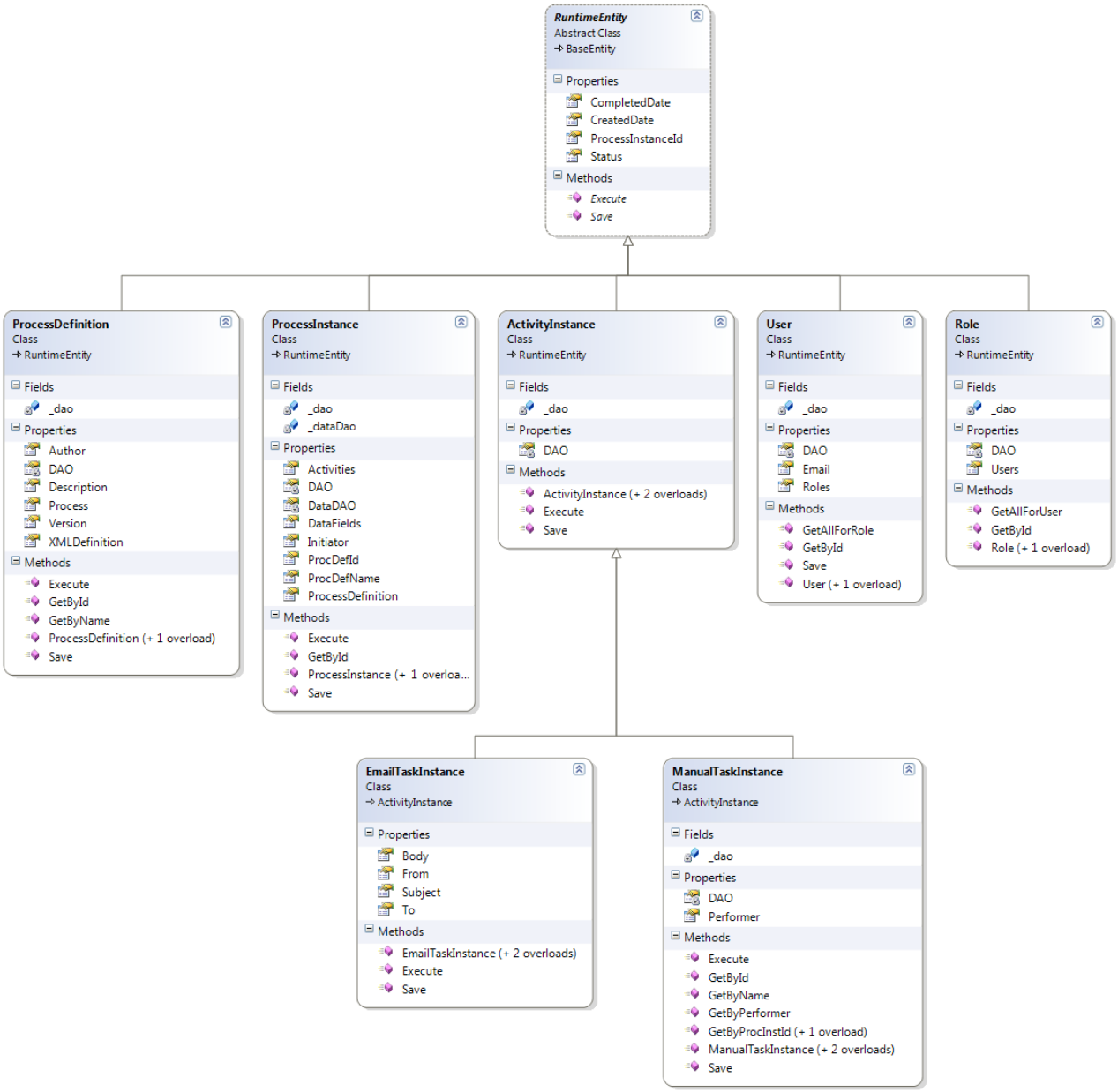

Figure 17: Runtime Entities (Lib.Entities.Runtime)

3.2.1 RuntimeEntity Class

The RuntimeEntity Class is an abstract class that inherits from EntityBase and adds runtime specific fields such as CreatedDate and CompletedDate. The RuntimeEntity class also defines the base methods that all Runtime Entities must override: Execute() and Save().

3.2.2 ProcessDefinition Class

The ProcessDefinition Class represents a versioned instance of a workflow definition. At runtime, an instance of this class is instantiated and used as a reference for the currently executing process instance.

3.2.3 ProcessInstance Class

The ProcessInstance class is the main runtime class. Instances of this class represent a single instance of a workflow. This class also contains references to the runtime activities for the workflow. After each activity is executed, the workflow engine uses the process definition referenced by the process instance to determine the next step in the workflow.

3.2.4 ActivityInstance Class

The ActivityInstance Class is the base class for which all runtime activities inherit from. This class is not abstract so functionality common to all activities is implemented in this class. For example, the event activity which does little more than create a record in the database when executed uses the Execute() method of this base class and does not override it.

3.2.5 EmailTaskInstance Class

The EmailTaskInstance Class is responsible for implementing an EmailTask as defined in the workflow's schema. This class sends the email and communicates with the data access layer to save its data.

3.2.6 ManualTaskInstance Class

The ManualTaskInstance Class represents a task that is not automatically completed by the workflow engine. Instances of this class represent a task assigned to a human participant and do not complete until manually told to by calling the CompleteTask() method in the workflow engine's main class.

3.3 Main Engine Logic

The main logic for the workflow engine is contained in the WFEngine class. I chose to implement most of the engine's functionality here so as not to make the definition and runtime entities larger with logic. Many instances of process entities are contained in memory at any given time so by not implementing much logic in them makes their memory footprint smaller. I wanted to consolidate the engine logic such that there was only one copy in memory at any given time.

3.3.1 WFEngine Class

As stated earlier, the WFEngine class holds most of the workflow engine's logic. It has five public methods for which to interact with a process instance: CreateProcessInstance, CancelProcessInstance, CompleteTask, GetDataField, and UpdateDataFields. Internally this class has private variables for the current process definition and process instance. These variables are used as local cache to improve performance by eliminating unnecessary trips to the database. Below is the class diagram for the WFEngine class.

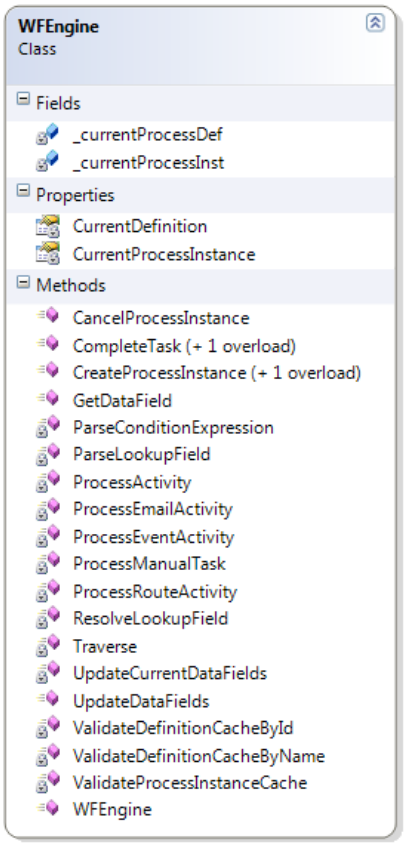

Figure 18: Main Engine Class (Lib.Engine)

3.4 Data Access Layer

All data access for this system is contained in a set of classes in the Lib.DAL namespace. Each runtime entity has a corresponding data access class which is responsible for retrieving and saving that type of object. All Classes of the Data Access Layer inherit from the DAOBase class. The DAOBase class defines the connection to the database. Figure 19 displays the full class diagram for the data access layer.

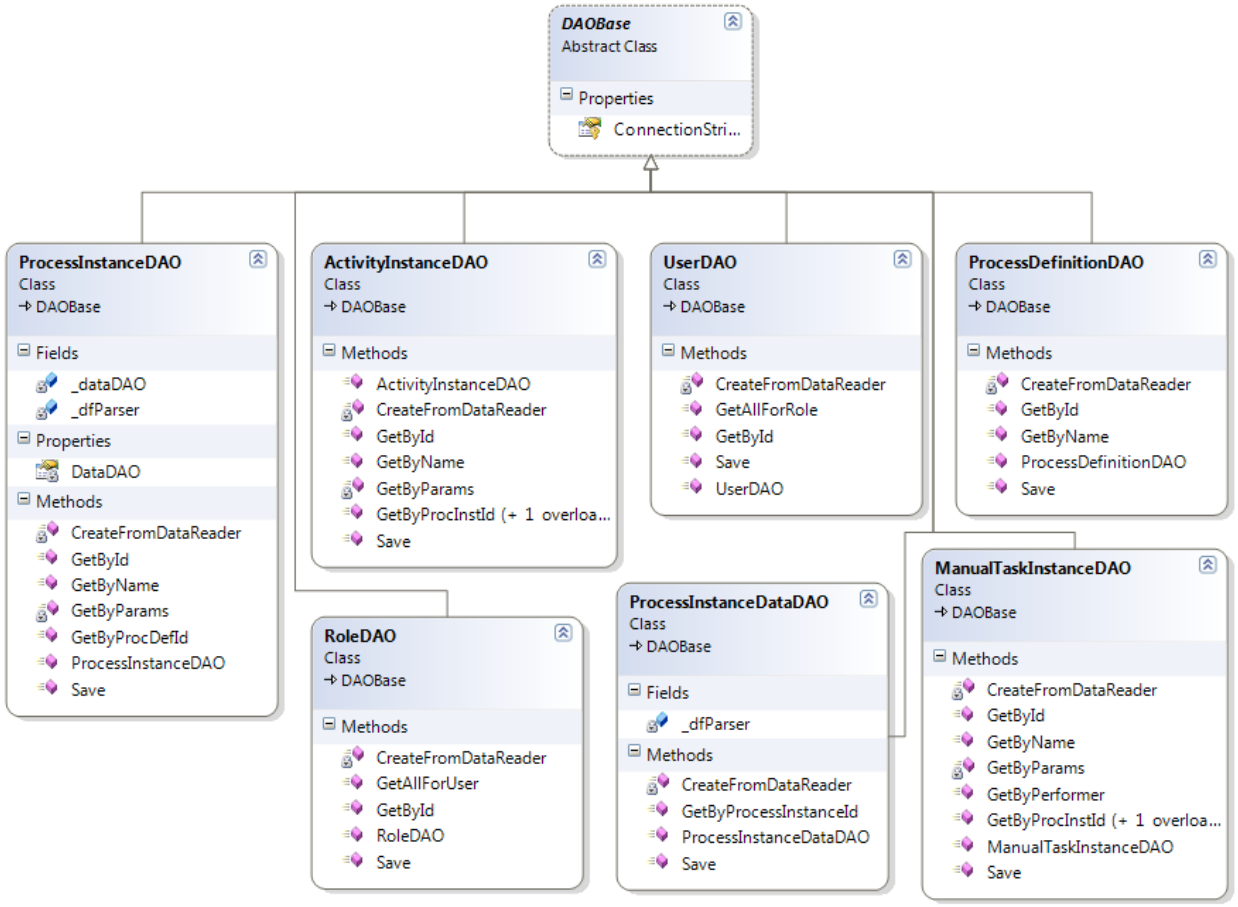

Figure 19: Data Access Layer (Lib.DAL)

3.4.1 DAOBase Class

The DAOBase class is the abstract base class that all data access classes inherit from. This class only defines the connection to the database.

3.4.2 ProcessDefinitionDAO Class

The ProcessDefinitionDAO class is responsible for retrieving and saving ProcessDefnintion objects. This class also uses the ProcDefXMLParser shared utility class (section 3.5.1) to parse the xml workflow definition schema from the database.

3.4.3 ProcessInstanceDAO Class

The ProcessInstanceDAO class is responsible for retrieving and saving ProcessInstance objects. This class also uses the ProcessInstaceDataDAO class (section 3.4.8) to retrieve the ProcessInstance classes list of DataFields.

3.4.4 ActivityInstanceDAO Class

The ActivityInstanceDAO class is responsible for retrieving and saving ActivityInstance objects.

3.4.5 UserDAO Class

The UserDAO class is responsible for retrieving and saving User objects.

3.4.6 RoleDAO Class

The RoleDAO class is responsible for retrieving and saving Role objects.

3.4.7 ManualTaskInstanceDAO

The ManualTaskInstanceDAO class is responsible for retrieving and saving ManualTaskInstance objects.

3.4.8 ProcessInstanceDataDAO

The ProcessInstanceDataDAO class is responsible for retrieving and saving ProcessInstance DataField objects in an xml format. This class uses the DataFieldsParser shared utility class (section 3.5.3) to parse the stored xml definition of the process instances data fields in a list of DataField objects.

3.5 Shared Utilities and Enumerations

The Lib.Shared namespace consists of two sub namespaces: Enums and Utils. The Enumerations in the Enums namespace represent the enumerations found in the XPDL schema such as Status and ParticipantType. The Operand Enumeration is the only custom enumeration and it is used by the ConditionEvaluator class (see section 3.5.2) to evaluate gateway conditions in the workflow. The Utils

namespace contains several classes that are used in the other namespaces to facilitate their specific purposes. For example, the ProcessDefinitionDAO class uses the ProcDefXMLParser class to fill a ProcessDefinition objects Process Property from the raw xml definition of the workflow. Below is the class diagram for both the Enums and Utils namespaces.

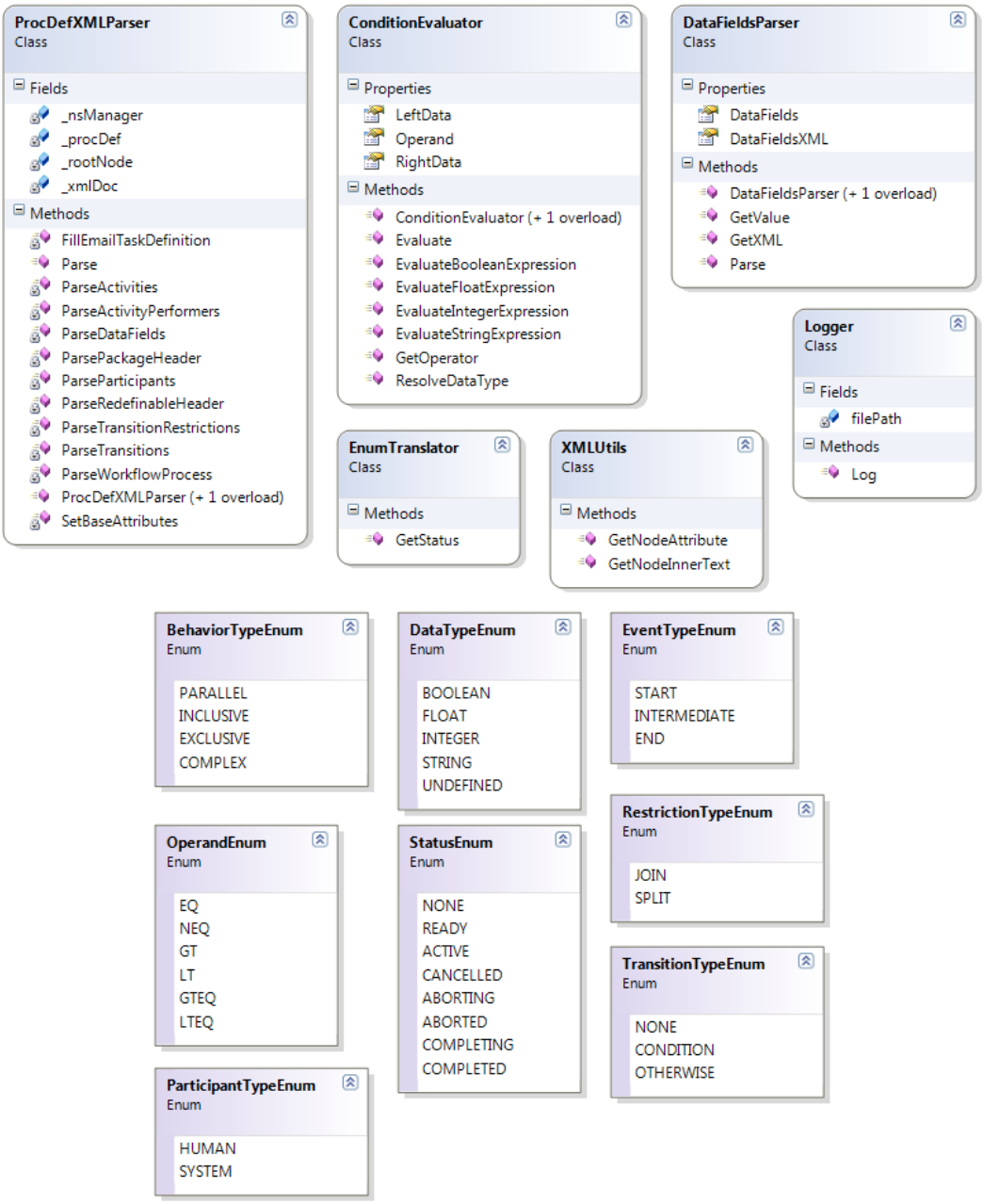

Figure 20: Enumerations and Shared Utilities

3.5.1 ProcDefXMLParser Class

The ProcDefXMLParser class is a very important class in this system. Its purpose is to parse the XPDL definition of a workflow into an object model that can be queried at runtime to traverse the workflows activities. Given a XPDL file, the ProcDefXMLParser class performs the following actions in its parse() method to parse the file and return the object model:

```
///<summary>
/// Parses the process definition xml and returns a process object
///</summary>
///<returns></returns>
publicProcess Parse()
    {
// Parse the package header
this.ParsePackageHeader();

// Parse the Redefinable header
this.ParseRedefinableHeader();

// Parse the Participants
this.ParseParticipants();

// Parse the workflow process
this.ParseWorkflowProcess();

// Parse the data fields
this.ParseDataFields();

// Parse the Activities
this.ParseActivities();

// Parse the transitions
this.ParseTransitions();

returnthis._procDef;
    }
```

Figure 21: ProcDefXMLParser Parse() Method

3.5.2 ConditionEvaluator Class

The ConditionEvaluator Class is responsible for parsing and evaluating an expression in the workflows definition. An example could be an expression like (${Amount} > 500). Given the previous expression, ${Amount} represents a lookup field that the ConditionEvaluator will have to resolve from the current process instances data field's collection. After pre-processing the data the ConditionEvaluator would then determine the operand type and the data types of both sides of the equation. The Evaluator would then evaluate the expression and return a Boolean result. Below are the two most important methods for this class: the Evaluate method that processes the expression and the ResovleDataType method which uses regular expressions to determine the data types of the expression's data fields.

```
///<summary>
/// Evaluates the dynamic expression
///</summary>
///<returns></returns>
publicbool Evaluate()
    {
if (this.LeftData != null&&this.RightData != null)
        {
if (this.LeftData.DataType == this.RightData.DataType)
          {
try
            {
switch (this.LeftData.DataType)
              {
caseDataTypeEnum.INTEGER:
return EvaluateIntegerExpression(Convert.ToInt32(this.LeftData.Value),
Convert.ToInt32(this.RightData.Value), this.Operand);
caseDataTypeEnum.BOOLEAN:
return EvaluateBooleanExpression(Convert.ToBoolean(this.LeftData.Value),
Convert.ToBoolean(this.RightData.Value), this.Operand);
caseDataTypeEnum.FLOAT:
return EvaluateFloatExpression(Convert.ToDecimal(this.LeftData.Value),
Convert.ToDecimal(this.RightData.Value), this.Operand);
caseDataTypeEnum.STRING:
return EvaluateStringExpression(this.LeftData.Value.ToString(), this.RightData.Value.ToString(),
this.Operand);
              }
            }
catch { }
          }
        }
returnfalse;
    }
```

Figure 22: ConditionEvaluator's Evaluate() Method

```
///<summary>
/// Resolves a data type
///</summary>
///<param name="value"></param>
///<returns></returns>
publicstaticDataTypeEnum ResolveDataType(string value)
    {
if (Regex.IsMatch(value, @"^\d+$"))
        {
returnDataTypeEnum.INTEGER;
        }
elseif (Regex.IsMatch(value, "^\\d+(\\.\\d+)?$"))
        {
returnDataTypeEnum.FLOAT;
        }
elseif (value.ToLower() == "true" || value.ToLower() == "false")
        {
returnDataTypeEnum.BOOLEAN;
        }
else
        {
returnDataTypeEnum.STRING;
        }
    }
```

Figure 23: ConditionEvaluator's ResolveDataType() Method

3.5.3 DataFieldsParser Class

The DataFieldsParser Class is used to extract a process instances data fields from their xml definition and serialize the data fields back to xml for persistence. Below is an example of the schema used when saving process instance variables at runtime.

```
<DataFields>
<DataFieldId=”_minp4FFwEeC7_uB68xeFUA"Name="
Amount"DisplayName="Amount"Type="INTEGER"Value="600"/>
<DataFieldId="_minp4FFwEeC7_uBwNhVvUA"Name="IsManagerApproved"DisplayName="Is Manager
Approved"Type="BOOLEAN"Value="true"/>
</DataFields>
```

Figure 24: DataField XML example

3.5.4 XMLUtils Class

The XMLUtils class provides a few common methods for processing xml nodes. This class is used to consolidate some common xml parsing tasks such extracting the value of an attribute or getting the inner text of a node.

3.5.5 EnumTranslator Class

The EnumTranslator Class is a centralized location for resolving an Enumeration to and from text. Currently the class only returns a status enumeration based on a text input.

3.5.6 Enumerations

The enumerations are shared throughout the various namespaces to reference those choices that are valid in the XPDL schema. The OperandEnum is the only custom enumeration and is used by the ConditionEvaluator class.

3.6 Execution Diagrams

This section contains three activity diagrams and one state diagram to illustrate how the workflow engine functions. The activity diagrams are broken down to illustrate the engine from three different perspectives. The first activity diagram (figure 25) shows all the activities that occur during the execution of a process instance. The second diagram (figure 26) shows all the activities associated with executing an activity. The third diagram (figure 27) shows all the activities associated with completing a manual task. The final diagram (figure 28) shows the overall states that a process instance can be in throughout its execution and the events that trigger each of these states.

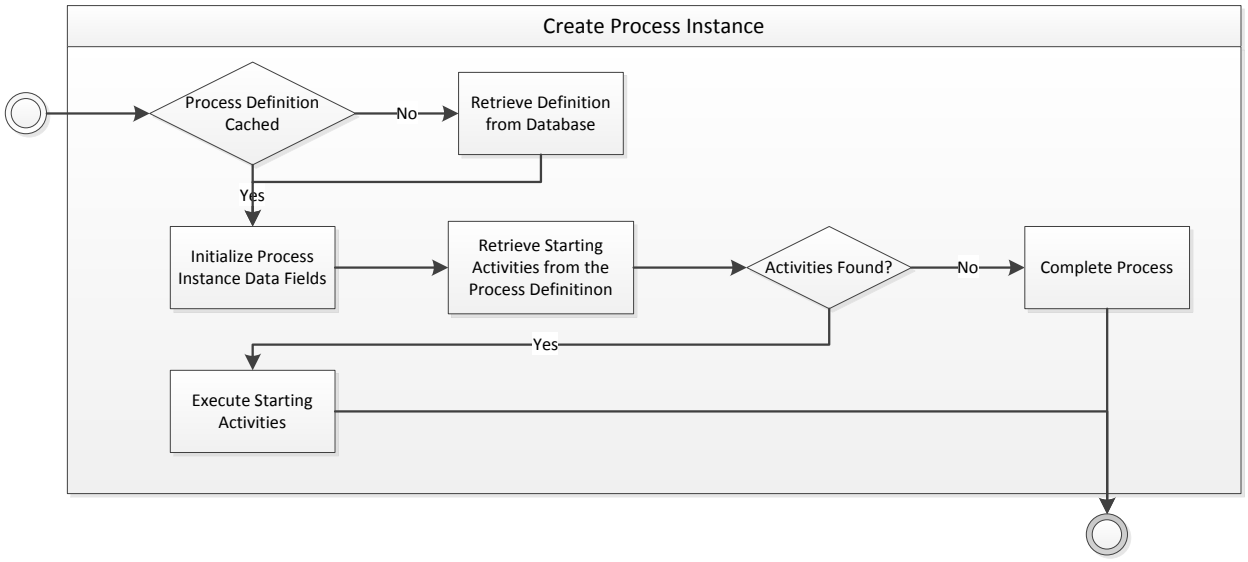

Figure 25: Creation of a Process Instance Activity Diagram

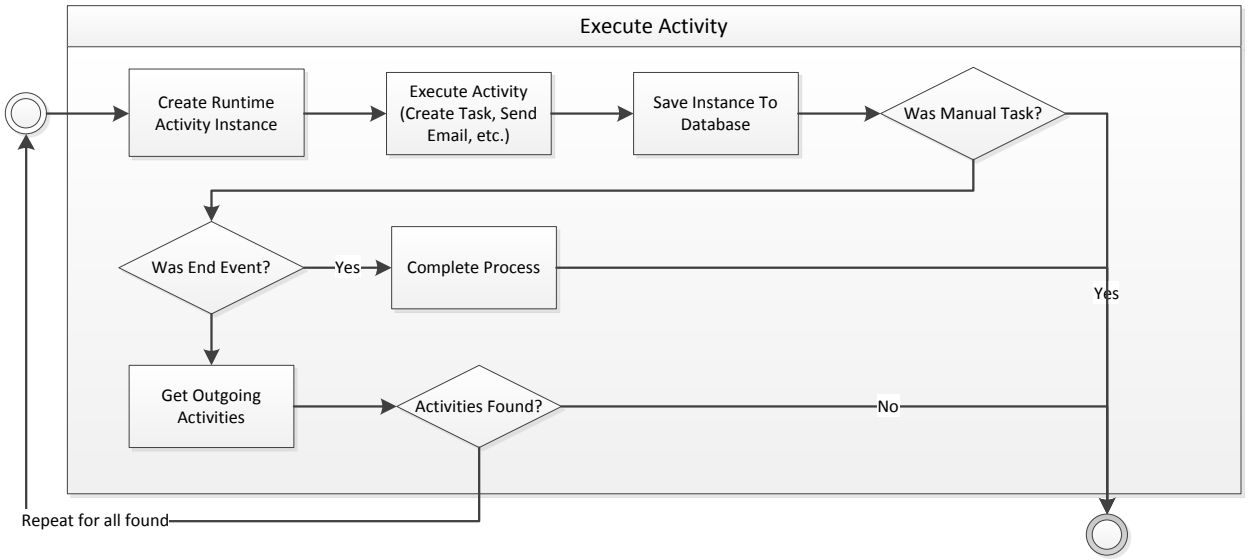

Figure 26: Executing an Activity, Activity Diagram

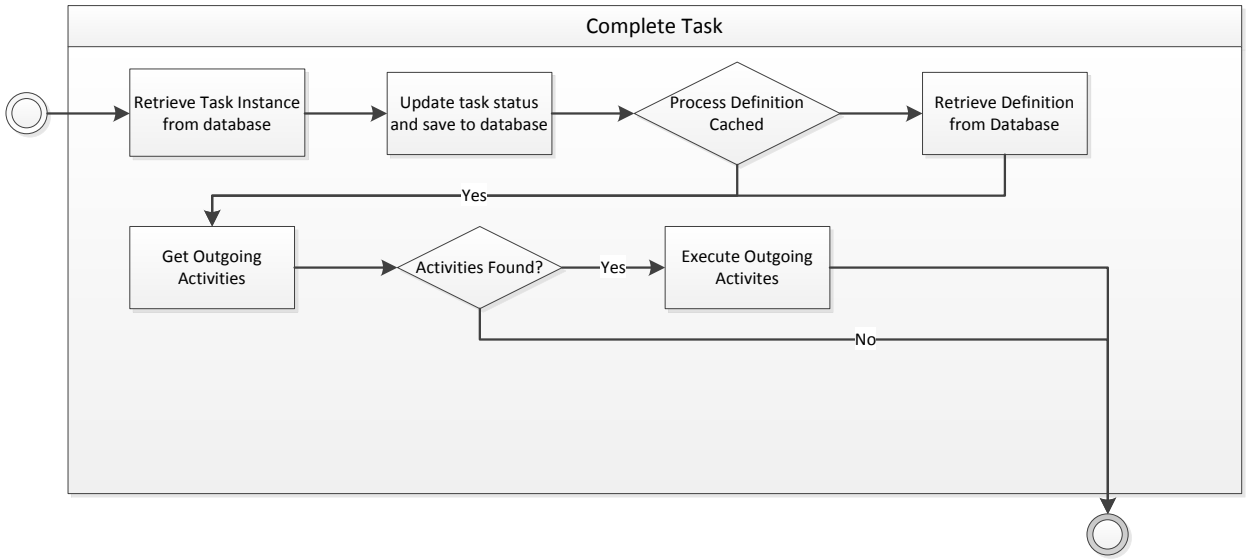

Figure 27: Completing a Manual Task Activity Diagram

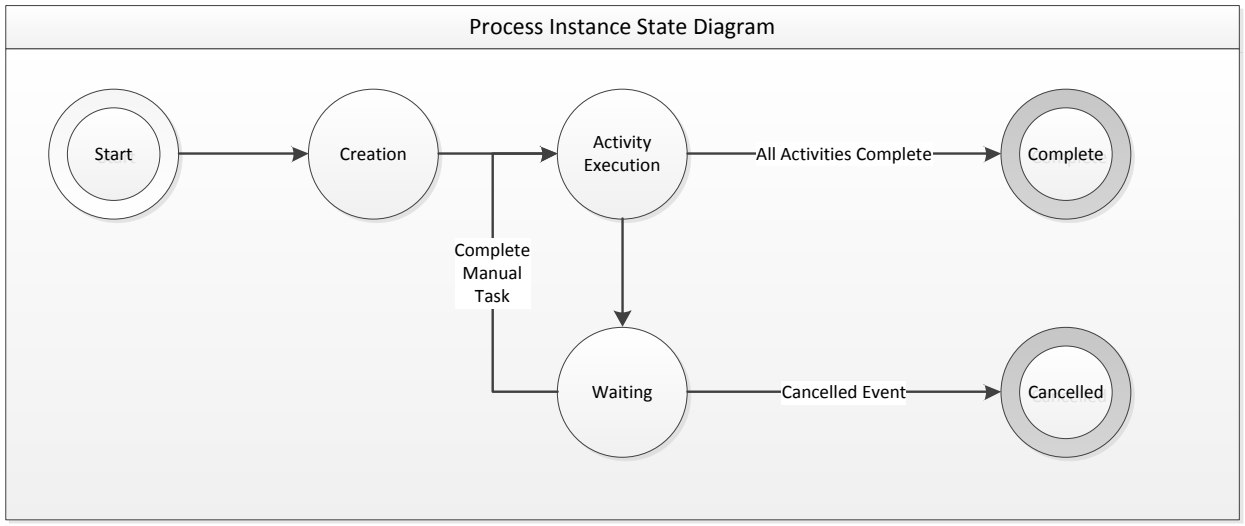

Figure 28: Process Instance State Diagram

4. Test Method

4.1 Hypotheses

Given a workflow management system developed on the .Net framework using the XPDL process standard; I propose that this application can be improved by separating concerns using aspect oriented software development techniques. More specifically, I propose that, by applying AOSD techniques to this system using the PostSharp AOP framework, cross-cutting concerns will become easier to add and manage and the number of lines of code will decrease in general. I also propose that the addition of AOSD techniques to the system will result in performance that is less than or equal to a pure object oriented approach. The research hypotheses are therefore:

Hypothesis 1: Applying AOSD techniques to a workflow management system using the PostSharp AOP framework will result in a decrease in the number of lines of code and decrease the amount of time needed to implement crosscutting concerns in the system.

Hypothesis 2: Applying AOSD techniques to a workflow management system using the PostSharp AOP framework will result in performance that is equal to that of the object oriented version in both execution time and memory usage.

4.2 Experiment 1: Addition of a Cross-Cutting Concern

For this experiment I added support for activity logging in both the aspect oriented and object oriented versions of the system. Specifically, I measured the difference in lines of code needed and the time to implement this feature between the two versions of the system. The main goal of this experiment was to prove hypothesis 1 which states that cross cutting concerns will be easier to add and result in fewer lines of code with Aspects.

4.2.1 Experiment 1: Test Results

For the first part of this experiment I wrote a Logger class which consisted of a single static method called Log() as seen in figure 29. The Log method takes a string message and a possible exception as input and logs them to a text file. The entire class only took about 10 minutes to write.

```
///<summary>
/// Logs to the the applications logfile
///</summary>
///<param name="message"></param>
publicstaticvoid Log(string message, System.Exception ex)
    {
StreamWriter logFile = newStreamWriter(filePath, true);
        logFile.WriteLine("{0}: {1}", DateTime.Now, message);
if (ex != null)
        {
           logFile.WriteLine(ex.ToString() + Environment.NewLine +
"<---------------------------------------------------------------------------------------->" + Environment.NewLine);
        }
        logFile.Close();
    }
```

Figure 29: The Log() Method

Once the Logger class was created I then went to each place where the Activity's Execute() method was implemented and added code similar to that in figure 30. Notice that the Log() method can be called up to three times every time an activity is executed: Once when the method begins, once on successful execution, and once if an exception if encountered.

```
///<summary>
/// Executes this activity and saves it to the database
///</summary>
publicoverridevoid Execute()
    {
try
        {
Logger.Log(this.DisplayName + " has started executing.", null);
this.Status = Shared.Enums.StatusEnum.COMPLETED;
this.CompletedDate = DateTime.Now;
// Saves the record in the database
this.Save();
Logger.Log(this.DisplayName + " has finished executing.", null);
        }
catch (System.Exception e)
        {
Logger.Log(this.DisplayName + " has failed to execute.", e);
        }
    }
```

Figure 30: Activity Execute with Logging Example

The addition of all the logging code took approximately 25 minutes to add to the object oriented version of the system. It needed to be added in five places.

For the Aspect Oriented version of the logging functionality I created an aspect called LoggingAspect which extended from the OnMethodBoundry Aspect type in PostSharp. I chose this Aspect type because the code it injects into the advised method is very similar to the code I added to each method in the object oriented version of the system. The Aspect took me approximately 15 minutes to write. The code for the aspect can be seen in figure 31. The aspect adds four events to the target method: OnEntry, OnException, OnSucess, and OnExit. I handled only the first three and added the same logging code as in the object oriented version.

```
///<summary>
/// This aspect is used to log whenever an activity is executed
///</summary>
   [Serializable]
publicsealedclassActivityLoggingAspect : OnMethodBoundaryAspect
   {
public ActivityLoggingAspect()
      {
this.AttributeInheritance = MulticastInheritance.Multicast;
      }

// Invoked at runtime if an execption occurs
publicoverridevoid OnException(MethodExecutionArgs args)
      {
Logger.Log(((RuntimeEntity)args.Instance).DisplayName + " has failed to execute.", args.Exception);
      }

// Invoked at runtime before that target method is invoked.
publicoverridevoid OnEntry(MethodExecutionArgs args)
      {
Logger.Log(((RuntimeEntity)args.Instance).DisplayName + " has started executing.", null);
      }

// Invoked at runtime after the target method is invoked (in a finally block).
publicoverridevoid OnSuccess(MethodExecutionArgs args)
      {
Logger.Log(((RuntimeEntity)args.Instance).DisplayName + " has finished executing.", null);
      }
   }

}
```

Figure 31: LoggingAspect

To apply this Aspect to all execute methods only took one line of code in the base runtime entity class. I added the attribute as seen in figure 32 to the abstract definition of the execute method as seen in figure 32.

```
      ///<summary>
/// Executes this activity
///</summary>
      [Aspects.ActivityLoggingAspect(AttributeInheritance = MulticastInheritance.Multicast)]
publicabstractvoid Execute();
```

Figure 32 Applying the Logging Aspect

After applying the logging functionality to both versions of the system I measured the results. For the object oriented version, logging took approximately 35 minutes to implement and resulted in 2,582 lines of code for the whole system. In contrast the Aspect oriented version took only 20 minutes to implement (I added the 10 minutes that the logging class took to initially develop). The final Aspect Oriented system resulted in 2,552 lines of code. Although in this experiment the difference in lines of code was only 30 lines (1% less code), it is easy to see that if scaled, the aspect oriented version would produce far less lines of code (like if there were 50 Activity types and therefore 50 execute methods rather than just 5). Not only would it produce less code the aspects would facilitate the maintenance of the crosscutting concerns in fewer places. It is based on these observations that I feel that hypothesis 1 is correct. Figure 33 shows a summary of this experiment.

	Time to implement	**Starting Lines of Code**	**total system lines of code after logging**	**Number of places needing code added**
OOWF	35 minutes	2502	2,582	5
AOWF	20 minutes	2502	2,552	2
Differences:	**15min**	**0**	**-30 lines or -1.2%**	**-3 places or -60%**

Figure 33: Addition of Logging Summary

4.3 Experiment 2: Efficiency

In this project I define Efficiency to mean the sum of total execution time and use of memory, i.e., a system with a smaller execution time and that uses less system memory will be considered more efficient. This experiment was not aimed at proving that the Aspect Oriented version of the workflow engine will be more efficient than the Object Oriented version but I did expect it to be either as efficient or slightly less. The main purpose of this experiment was to justify that Aspect Oriented techniques do not add significant overhead as compared to standard object oriented approaches from an efficiency standpoint.

For this experiment I automated and ran a test workflow three times for both versions of the system. The test workflow was the expense approval workflow outlined in appendix A and I've configured the test to pass through both levels of approval. In between each run I cleared the database and cached all remaining records using select statements. I did this in between each run to minimize the disk I/O when accessing the database. Specifically, I measured and averaged the execution time down to the method level using the ANTS performance profiler for both versions of the system and compared the execution times. After completing the execution time comparison I re-ran the scenario and compared the use of system memory for both versions of the system (ANTS was not able to do performance testing and memory usage at the same time).

4.3.1 Test Process Definition

For this test I useda process model designed in Business Studio and exported into the XPDL format. The model represents a simple expense approval workflow in which an expense goes through one or two levels of approval based on the dollar amount of the expense. During execution of the workflow tasks, email messages, and conditional gateways were utilized to complete the process. See figures 34 for the process model and appendix A for the full XPDL schema.

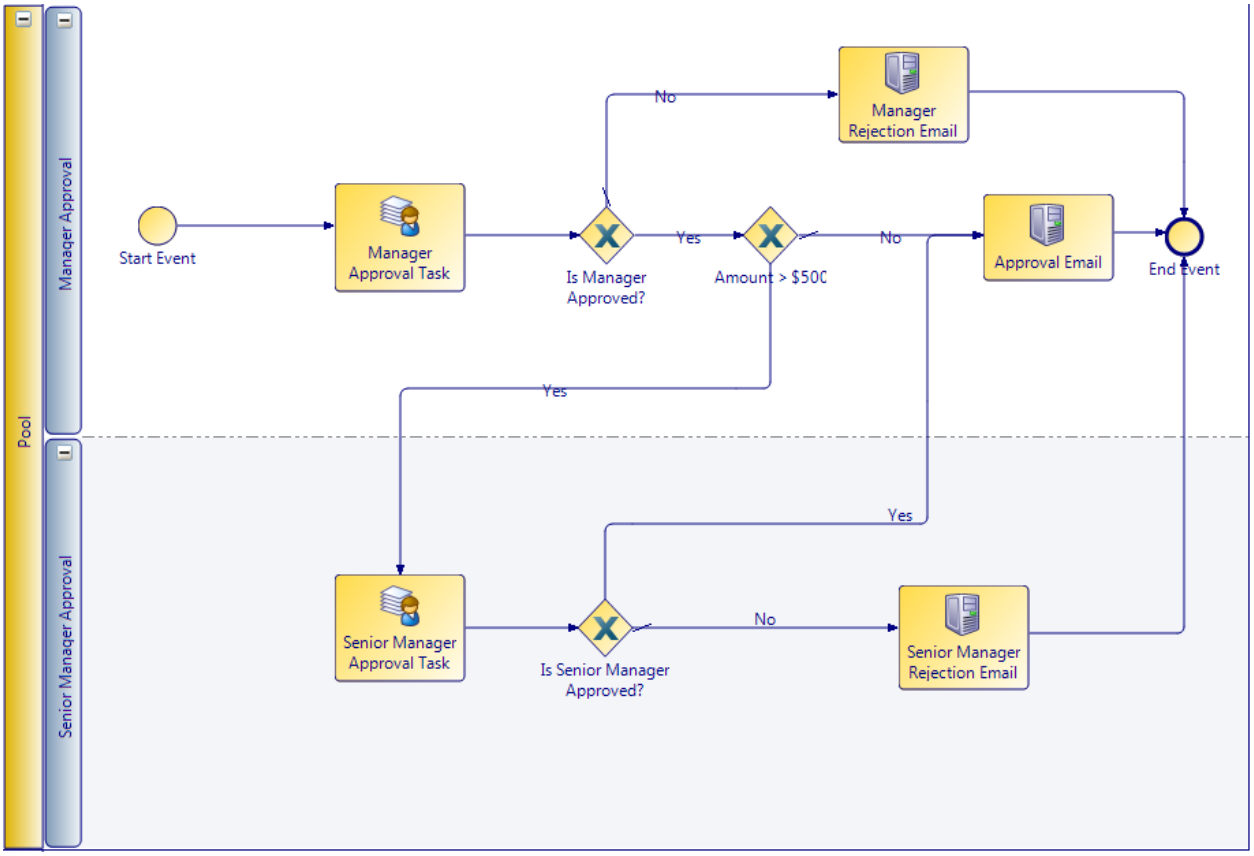

Figure 34: Expense Approval Process

4.3.2Test Results

After running the performance tests on both the object oriented and aspect oriented versions of the engine I was able to confirm my hypothesis that the aspect oriented version was negligibly slower on average with and without logging enabledthan the object oriented version. The Aspect Oriented version performed 1.46% or 318 milliseconds slower with logging and 1.48% or 322 milliseconds without logging.I attribute this difference to fluctuations in disk I/O speed and external processes needing the CPU. In both test cases the logging function was called twelve times per version of the system. The results were not surprising because the injected code from the logging aspect was almost identical to that of the object oriented version. Figure 35 showsan example of the decompiled code.

```
[HasInheritedAttribute(new long[] { -8517338228674527231L })]
public override void Execute()
{
MethodExecutionArgs CS$0$0__aspectArgs = new MethodExecutionArgs(this, null);
<>z__Aspects.a0.OnEntry(CS$0$0__aspectArgs);
   try
   {
      base.Status = StatusEnum.COMPLETED;
      base.CompletedDate = new DateTime?(DateTime.Now);
      this.Save();
<>z__Aspects.a0.OnSuccess(CS$0$0__aspectArgs);
   }
   catch (Exception CS$0$1__exception)
   {
      CS$0$0__aspectArgs.Exception = CS$0$1__exception;
<>z__Aspects.a0.OnException(CS$0$0__aspectArgs);
      throw;
   }
}
```

Figure 35: Decompiled ActivityInstance Execute Method with applied aspect

Below are the summary of the runs and a summary of the logging performance. For all performance data see appendix B.

	All times in Milliseconds				
	Run 1	**Run 2**	**Run 3**	**Total**	**Average**
OOWF	21567.537	21408.846	21514.888	64690.877	21563.626
AOWF	21767.143	21646.720	22031.628	65445.491	21815.164
% difference:	+.917%	+1.099%	+2.345%	+1.458%	**+1.458%**

Figure 36: Performance with Logging

	All times in Milliseconds				
	Run 1	**Run 2**	**Run 3**	**Total**	**Average**
OOWF	21487.528	21329.475	21451.519	64268.522	21422.841
AOWF	21692.807	21573.343	21968.020	65234.170	21744.723
% difference	+.946%	+1.130%	+2.351%	+1.480%	**+1.480%**

Figure 37:Performance without Logging

After running the memory usage tests against both versions of the system I was also able to confirm that the aspect oriented version of the system used slightly more system memory than the object oriented version (approximately 1.8%). I also showed that the Aspect Oriented version had an

average stack size that was about 6% greater than the object oriented version. I suspect that this has something to do with the fact that the logging aspect added another class to the system which would need to be loaded at runtime. Below are the summaries for the memory usage results and the stack heights for each run along with their averages.

	All results are in kilobytes and reflect the averages at each step and the total average				
	At start	**After Process Creation**	**After Manager Approval**	**After Senior Manager Approval**	**Average Memory Used**
OOWF	411.237	930.364	997.429	1182.579	880.402
AOWF	412.620	967.747	1034.812	1169.962	896.285
% difference	+0.335kb	+3.863kb	+3.613kb	-1.078kb	**+1.772%**

Figure 38: Memory Usage Summary

	Stack Height				
	Run 1	**Run 2**	**Run 3**	**Total**	**Average**
OOWF	127	127	128	382	127.333
AOWF	135	135	137	407	135.667
% difference	+5.926%	+5.926%	+6.569%	+6.143%	**+6.143%**

Figure 39: Stack Height Summary

5. Discussion and Future Work

5.1 Project Rationale

The reason why I chose this project as opposed to others was that I wanted to gain some valuable experience in the area of workflow management. Since workflow management systems play such an integral role in many industries (including my own in manufacturing), finding a better way to improve their implementation seemed like a worthwhile task. Also, it was my hope that this project would increase my understanding of Aspect Oriented Software Development in order to determine its feasibility in everyday use. The workflow engine itself was very challenging for me to implement and there are admittedly many places where improvements could still be made. In hindsight, I am still glad that I chose this project because I now have a much better understanding of Workflow Management Systems, XPDL, and Aspect Oriented Software Development.

5.2 Results Interpretation

After running the tests, the results I obtained helped me to see the benefit of using aspects. Both the memory usage and performance seemed to be on par with the object oriented version and adding the logging concern was much easier using the aspect oriented version. In a world where the common saying in software development is “If I only had more time” and refactoring is just wishful thinking, the aspect oriented approach could allow a developer to spend more time on more difficult functionality. Of course everything comes with a price. It seems that what you gain in development time could, if not managed correctly, lead to a maintenance nightmare. At least with the object oriented version you can see the code as it will be when it is executed. If you don’t know how the aspects are applied then debugging a system could prove to be extremely difficult. Therefore, the choice to use aspects should not be taken lightly. For large applications it can be overwhelming but can

also be a very powerful tool. In my opinion, aspects should only be used cautiously but I cannot deny their potential.

5.3 Strengths and Weaknesses

5.3.1 Workflow Engine

As with any system there will be strengths and weaknesses. For my system, I feel that its best features include the adherence to the XPDL schema and lightweight design. With the object model I tried as much as possible to adhere to the schema (even with its nuances). Unfortunately I was not able to fully support the entire schema due to time constraints but I was able to model most of the core components. The lightweight design refers to most of the engines' logic being centrally located in the WFEngine class. The reason I felt that this was a good thing is because many of the objects are being used and cached in memory so I did not want to load them with a lot of logic. My goal was to maximize system memory usage to account for a potentially high volume of transactions.

The main weaknesses of my system are its lack of asynchronicity (being able to process expressions or execute activities in parallel), missing support for some XPDL elements, and its simplified expression processing. The lack of asynchronicity is due to the complex nature of the system and the time it took to develop. I wanted to understand the engine before trying to deal with asynchronous requests. The missing XPDL elements mostly include support for sub-processes. I plan to incorporate this functionality in future revisions. As for the expression processing, I did not feel that making a fully functional expression processor utility would have added a lot of value to the experiments but it is something that the final version of this system will need to have. Each of these weaknesses was a calculated decision that I made in the interest of time. As with any project, with the right amount of time and effort, anything is possible.

5.3.2 Use of Aspects

The decision to use aspects is a difficult one for me to make. They seem to be great as a time savings or for adding functionality to applications in which source code is not available. On the other hand, they can make debugging an application more difficult because the concerns are separated and you can't "see" what is going to happen just by looking at an advised method. Given the choice, I would only use aspects when refactoring is not an option or if refactoring does not give me the level of concern separation I want.

Another issue when dealing with aspects is functional requirements. By design, aspects are not meant for functional requirements but in practice this is difficult to do. For example, when implementing the logging aspect in this project I was faced with a small problem. I needed to log a specific message for a specific activity type. To do this in the object oriented version of the system was easy, I just wrote the message in the function call. In the Logging Aspect I had to either use reflection to determine the activity type or just log a generic message. In doing so I needed to add functional requirements to the aspects. I think in general this is ok if kept to a minimal but this can lead to a lot of functional requirements being implemented in aspects. I think the key take away here is that aspects are not a perfect solution. The decision to use them must be calculated based on the task at hand.

5.4 Challenges Encountered

There were two main challenges that I encountered during this project. The first dealt with deciphering the XPDL schema. The schema itself was not very hard to understand but I found the way that some of the elements were structured to be frustrating. For example, the XPDL definition of an activity in the workflow does not contain an attribute at the root level to let the interpreting program know what type of activity with which it is dealing with. To determine the type of activity, I had to search the child nodes of an activity using XPath queries to check for the existence of certain nodes and

attributes. This seemed like it was causing the engine to do extra processing to make type determinations. Another example of this is in the way that the transitions were stored. I felt that the transitions between activities would have been more useful in the Activity's definition rather than in its own collection. This also forced the parsing program to perform extra work when constructing the object model.

The second main challenge I encountered during this project was processing the lookup data and the free text expressions used to control the sequence of activities in the workflow. TIBCO's Business Studio did not provide any method of creating or validating conditional expressions within the process definition. Due to this, all conditions in the test process were in free text. To further complicate things, I chose to support the use of key fields in the definition. Key fields refer to place holders in the process definition that will need to be resolved at runtime. An example of this would be where the amount is verified to determine how many approvals are needed. This expression form looked like "$DATA{Amount} > 500". The syntax "$DATA{}" told the engine that this field needs to be filled in at runtime with the value in the process instances data fields with the name "Amount". On top of this I needed to parse and evaluate the expression into a Boolean. To do this I had to only support expressions in the form DATA Operand DATA. This is of course not an ideal solution and is a topic for further research.

5.5 Future Work

Given more time and resources, there are several things I would like to explore in more detail. The first thing I would like to examine is what effect new requirements would have on an aspect oriented system. For example, let's say that you wanted to change the behavior of the execute method for an email task but not for a manual task. This scenario is good because it raises a couple of interesting questions like "do I need a totally new aspect or can I just modify the existing aspect?"

Another question in this scenario would be “What if there was more than one aspect applied to the method?” and how would that effect the situation.

The next thing I would like to take a look at is how well aspects can be applied to non-crosscutting concerns. Using the workflow engine as an example I would like to place the activity execution code found in the runtime entities into aspects. The reason I would like to do this is to try and maintain a workflow engine in which most of the logic is injected using aspects. This may or may not be an ideal solution but I think it would be interesting to try.

The last thing I would like to do is refactor the engine in order to support asynchronous requests and more of the XPDL schema. The current implementation does not support the entire schema nor does it support asynchronous requests. This was mostly due to time constraints. I also wanted to make sure that I understood the schema and how the engine would traverse it without adding the complexity of asynchronicity.

6. Bibliography

1. Mohan, C. (n.d.). *Recent Trends in Workflow Management Products, Standards, and Research*
2. Graham, B. (2008). *The Roots of the Business Process Mapping.*
3. http://office.microsoft.com/en-us/visio/
4. http://www.smartdraw.com/product/
5. http://www.conceptdraw.com/en/products/cdoffice/main.php
6. GregorKiczales, John Lamping, AnuragMendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, John Irwin, (1997), *Aspect-Oriented Programming*
7. Center for Technology in Government, University at Albany / SUNY, (1997), *An Introduction to Workflow Management Systems*
8. Robert M. Shapiro, (2008), *XPDL 2.1 Integrating Process Interchange and BPMN*
9. http://www.tibco.com/products/bpm/process-modeling/default.jsp
10. http://www.sharpcrafters.com/postsharp
11. ISO 12052:2006
12. http://en.wikipedia.org/wiki/Workflow
13. http://www.wfmc.org/
14. http://en.wikipedia.org/wiki/Aspect-oriented_software_development
15. Dr. a. Mark Doggett, (2004), *Selected Collaborative Problem-Solving Methods for Industry*
16. The Ben Graham Corporation, (1999), *Business Process Improvement Methodology*
17. Carnegie Mellon University, (2003), *Software Quality Attributes and Quality Tradeoffs*
18. Carnegie Mellon University, (1995), *Quality Attributes*
19. Microsoft Lines of Code Counter, http://code.msdn.microsoft.com/LOCCounter

20. ANTs Performance Profiler, http://www.red-gate.com/products/dotnet development/ants-performance-profiler/
21. The Workflow Management Coalition, (2008), *"Process Definition Interface – XML Process Definition Language"*

Appendix A: Expense Approval Workflow XPDL

```
<?xmlversion="1.0"encoding="UTF-8"?>
<xpdl2:Packagexmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"xmlns:database="http://www.tibco.com/XPD/database1.0.0"
xmlns:eaijava="http://www.tibco.com/XPD/EAIJava1.0.0"
xmlns:email="http://www.tibco.com/XPD/email1.0.0"
xmlns:iProcessExt="http://www.tibco.com/XPD/iProcessExt1.0.0"
xmlns:orchestrator="http://www.tibco.com/XPD/orchestrator1.0.0"
xmlns:order="http://www.tibco.com/XPD/order1.0.0"
xmlns:simulation="http://www.tibco.com/xpd/Simulation1.0.1"
xmlns:xpdExt="http://www.tibco.com/XPD/xpdExtension1.0.0"
xmlns:xpdl2="http://www.wfmc.org/2008/XPDL2.1"
xsi:schemaLocation="http://www.wfmc.org/2008/XPDL2.1
http://www.wfmc.org/standards/bpmnxpdl_31.xsd"Id="__EgBEFFsEeC7_uBwNhVvUA"
xpdExt:DisplayName="ExpenseApproval"Name="ExpenseApproval">
<xpdl2:PackageHeaderxpdExt:Language="en_US">
<xpdl2:XPDLVersion>2.1</xpdl2:XPDLVersion>
<xpdl2:Vendor>TIBCO</xpdl2:Vendor>
<xpdl2:Created>2011-03-18</xpdl2:Created>
<xpdl2:Description></xpdl2:Description>
<xpdl2:Documentation></xpdl2:Documentation>
<xpdl2:CostUnit>USD</xpdl2:CostUnit>
</xpdl2:PackageHeader>
<xpdl2:RedefinableHeaderPublicationStatus="UNDER_REVISION">
<xpdl2:Author>PeloquinEJ</xpdl2:Author>
<xpdl2:Version>1.0.0.0</xpdl2:Version>
</xpdl2:RedefinableHeader>
<xpdl2:Participants>
<xpdl2:ParticipantId="_2fjvEFFtEeC7_uBwNhVvUA"xpdExt:DisplayName="Manager"Name="$ROLE{Man
ager}">
<xpdl2:ParticipantTypeType="HUMAN"/>
</xpdl2:Participant>
<xpdl2:ParticipantId="_4GhF0FFtEeC7_uBwNhVvUA"xpdExt:DisplayName="Senior
Manager"Name="$ROLE{Senior Manager}">
<xpdl2:ParticipantTypeType="HUMAN"/>
</xpdl2:Participant>
<xpdl2:ParticipantId="_WxCmYFV0EeCITe_kgVEprw"xpdExt:DisplayName="Process
Initiator"Name="${ProcessInitiator}">
<xpdl2:ParticipantTypeType="HUMAN"/>
</xpdl2:Participant>
<xpdl2:ParticipantId="_C20eMlV1EeCITe_kgVEprw"xpdExt:DisplayName="System
Account"Name="SystemAccount">
<xpdl2:ParticipantTypeType="SYSTEM"/>
<xpdExt:SharedResourceType="Email"Name="EmailSender"/>
</xpdl2:Participant>
```

```
</xpdl2:Participants>
<xpdl2:Pools>
<xpdl2:PoolId="_UMtJ51FtEeC7_uBwNhVvUA"xpdExt:DisplayName="Pool"Name="Pool"BoundaryVisibl
e="true"Process="__EpyEFFsEeC7_uBwNhVvUA">
<xpdl2:Lanes>
<xpdl2:LaneId="_UMtJ6FFtEeC7_uBwNhVvUA"xpdExt:DisplayName="Manager
Approval"Name="ManagerApproval">
</xpdl2:Lane>
<xpdl2:LaneId="_dMnEIFFtEeC7_uBwNhVvUA"xpdExt:DisplayName="Senior Manager
Approval"Name="SeniorManagerApproval">
</xpdl2:Lane>
</xpdl2:Lanes>
</xpdl2:Pool>
</xpdl2:Pools>
<xpdl2:WorkflowProcesses>
<xpdl2:WorkflowProcessId="__EpyEFFsEeC7_uBwNhVvUA"xpdExt:DisplayName="Expense Approval
Process"Name="ExpenseApprovalProcess">
<xpdl2:ProcessHeader>
<xpdl2:Description></xpdl2:Description>
</xpdl2:ProcessHeader>
<xpdl2:DataFields>
<xpdl2:DataFieldId="_minp4FFwEeC7_uBwNhVvUA"xpdExt:DisplayName="IsManagerApproved"Name=
"IsManagerApproved">
<xpdl2:DataType>
<xpdl2:BasicTypeType="BOOLEAN"/>
</xpdl2:DataType>
<xpdExt:InitialValues>
<xpdExt:Value>false</xpdExt:Value>
</xpdExt:InitialValues>
</xpdl2:DataField>
<xpdl2:DataFieldId="_v2UC0FFwEeC7_uBwNhVvUA"xpdExt:DisplayName="IsSeniorManagerApproved"
Name="IsSeniorManagerApproved">
<xpdl2:DataType>
<xpdl2:BasicTypeType="BOOLEAN"/>
</xpdl2:DataType>
<xpdExt:InitialValues>
<xpdExt:Value>false</xpdExt:Value>
</xpdExt:InitialValues>
</xpdl2:DataField>
<xpdl2:DataFieldId="_g7AGsFV5EeCITe_kgVEprw"xpdExt:DisplayName="Amount"Name="Amount">
<xpdl2:DataType>
<xpdl2:BasicTypeType="FLOAT">
<xpdl2:Precision>10</xpdl2:Precision>
<xpdl2:Scale>2</xpdl2:Scale>
</xpdl2:BasicType>
</xpdl2:DataType>
</xpdl2:DataField>
</xpdl2:DataFields>
```

```
<xpdl2:Activities>
<xpdl2:ActivityId="_UMtJ6VFtEeC7_uBwNhVvUA"Name="StartEvent"xpdExt:DisplayName="Start
Event">
<xpdl2:Event>
<xpdl2:StartEventTrigger="None"/>
</xpdl2:Event>
</xpdl2:Activity>
<xpdl2:ActivityId="_UMtJ6lFtEeC7_uBwNhVvUA"Name="EndEvent"xpdExt:DisplayName="End Event">
<xpdl2:Event>
<xpdl2:EndEventResult="None"/>
</xpdl2:Event>
<xpdl2:TransitionRestrictions>
<xpdl2:TransitionRestriction>
<xpdl2:JoinType="Exclusive"ExclusiveType="Data"/>
</xpdl2:TransitionRestriction>
</xpdl2:TransitionRestrictions>
</xpdl2:Activity>
<xpdl2:ActivityId="_iMnaUFFtEeC7_uBwNhVvUA"Name="ManagerApprovalTask"IsATransaction="false"
xpdExt:Visibility="Private"xpdExt:DisplayName="Manager Approval
Task"xpdExt:RequireNewTransaction="false">
<xpdl2:Implementation>
<xpdl2:Task>
<xpdl2:TaskManual>
<xpdl2:Performers>
<xpdl2:Performer>-defined-in-Activity-Performer-</xpdl2:Performer>
</xpdl2:Performers>
</xpdl2:TaskManual>
</xpdl2:Task>
</xpdl2:Implementation>
<xpdl2:Performers>
<xpdl2:Performer>_2fjvEFFtEeC7_uBwNhVvUA</xpdl2:Performer>
</xpdl2:Performers>
<xpdl2:Extensions/>
<xpdExt:AssociatedParameters>
<xpdExt:AssociatedParameterFormalParam="IsManagerApproved"Mode="INOUT"Mandatory="false">
<xpdl2:Description></xpdl2:Description>
</xpdExt:AssociatedParameter>
</xpdExt:AssociatedParameters>
<xpdExt:ActivityResourcePatterns>
<xpdExt:WorkItemPriorityInitialPriority="50"/>
</xpdExt:ActivityResourcePatterns>
</xpdl2:Activity>
<xpdl2:ActivityId="_tFUpQFFtEeC7_uBwNhVvUA"Name="SeniorManagerApprovalTask"IsATransaction="
false"xpdExt:Visibility="Private"xpdExt:DisplayName="Senior Manager Approval
Task"xpdExt:RequireNewTransaction="false">
<xpdl2:Implementation>
<xpdl2:Task>
<xpdl2:TaskManual>
```

```
<xpdl2:Performers>
<xpdl2:Performer>-defined-in-Activity-Performer-</xpdl2:Performer>
</xpdl2:Performers>
</xpdl2:TaskManual>
</xpdl2:Task>
</xpdl2:Implementation>
<xpdl2:Performers>
<xpdl2:Performer>_4GhF0FFtEeC7_uBwNhVvUA</xpdl2:Performer>
</xpdl2:Performers>
<xpdl2:Extensions/>
<xpdExt:AssociatedParameters>
<xpdExt:AssociatedParameterFormalParam="IsSeniorManagerApproved"Mode="INOUT"Mandatory="fa
lse">
<xpdl2:Description></xpdl2:Description>
</xpdExt:AssociatedParameter>
</xpdExt:AssociatedParameters>
<xpdExt:ActivityResourcePatterns>
<xpdExt:WorkItemPriorityInitialPriority="50"/>
</xpdExt:ActivityResourcePatterns>
</xpdl2:Activity>
<xpdl2:ActivityId="__Q4awFFtEeC7_uBwNhVvUA"Name="Amount500"xpdExt:DisplayName="Amount >
$500">
<xpdl2:RouteGatewayType="Exclusive"MarkerVisible="true"ExclusiveType="Data"/>
<xpdl2:TransitionRestrictions>
<xpdl2:TransitionRestriction>
<xpdl2:SplitType="Exclusive"ExclusiveType="Data">
<xpdl2:TransitionRefs>
<xpdl2:TransitionRefId="_GZnYAFFuEeC7_uBwNhVvUA"/>
<xpdl2:TransitionRefId="_IlUakFFuEeC7_uBwNhVvUA"/>
</xpdl2:TransitionRefs>
</xpdl2:Split>
</xpdl2:TransitionRestriction>
</xpdl2:TransitionRestrictions>
</xpdl2:Activity>
<xpdl2:ActivityId="_dRqNAVVvEeCITe_kgVEprw"Name="IsManagerApproved"xpdExt:DisplayName="Is
Manager Approved?">
<xpdl2:RouteGatewayType="Exclusive"MarkerVisible="true"ExclusiveType="Data"/>
<xpdl2:TransitionRestrictions>
<xpdl2:TransitionRestriction>
<xpdl2:SplitType="Exclusive"ExclusiveType="Data">
<xpdl2:TransitionRefs>
<xpdl2:TransitionRefId="_fDunkVVvEeCITe_kgVEprw"/>
<xpdl2:TransitionRefId="_ZqKscFVwEeCITe_kgVEprw"/>
</xpdl2:TransitionRefs>
</xpdl2:Split>
</xpdl2:TransitionRestriction>
</xpdl2:TransitionRestrictions>
</xpdl2:Activity>
```

```
<xpdl2:ActivityId="_xrOHIFVvEeCITe_kgVEprw"Name="IsSeniorManagerApproved"xpdExt:DisplayName
="Is Senior Manager Approved?">
<xpdl2:RouteGatewayType="Exclusive"MarkerVisible="true"ExclusiveType="Data"/>
<xpdl2:TransitionRestrictions>
<xpdl2:TransitionRestriction>
<xpdl2:SplitType="Exclusive"ExclusiveType="Data">
<xpdl2:TransitionRefs>
<xpdl2:TransitionRefId="_25_SE1VvEeCITe_kgVEprw"/>
<xpdl2:TransitionRefId="_9Dj4YFVvEeCITe_kgVEprw"/>
</xpdl2:TransitionRefs>
</xpdl2:Split>
</xpdl2:TransitionRestriction>
</xpdl2:TransitionRestrictions>
</xpdl2:Activity>
<xpdl2:ActivityId="_wtGAEFVxEeCITe_kgVEprw"Name="SeniorManagerRejectionEmail"IsATransaction="
false"xpdExt:Visibility="Private"xpdExt:DisplayName="Senior Manager Rejection
Email"xpdExt:RequireNewTransaction="false">
<xpdl2:Implementation>
<xpdl2:Task>
<xpdl2:TaskServicexpdExt:ImplementationType="E-Mail"Implementation="Other">
<xpdl2:MessageInId="_aLxycFV1EeCITe_kgVEprw"/>
<xpdl2:MessageOutId="_aLxycVV1EeCITe_kgVEprw"/>
<email:Email>
<email:Definition>
<email:Fromemail:Configuration="Server"/>
<email:To>${ProcessInitiator}</email:To>
<email:Subject>Your Expense Report has been rejected by your senior manager</email:Subject>
</email:Definition>
<email:Body>Your Expense Report has been rejected by your senior manager</email:Body>
<email:SMTPemail:Configuration="Server"/>
</email:Email>
</xpdl2:TaskService>
</xpdl2:Task>
</xpdl2:Implementation>
<xpdl2:Performers>
<xpdl2:Performer>_C20eMlV1EeCITe_kgVEprw</xpdl2:Performer>
</xpdl2:Performers>
<xpdl2:Extensions/>
<xpdExt:ActivityResourcePatterns>
<xpdExt:AllocationStrategyxpdExt:Strategy="SYSTEM_DETERMINED"/>
</xpdExt:ActivityResourcePatterns>
</xpdl2:Activity>
<xpdl2:ActivityId="_CyeREVVyEeCITe_kgVEprw"Name="ManagerRejectionEmail"IsATransaction="false"
xpdExt:Visibility="Private"xpdExt:DisplayName="Manager Rejection
Email"xpdExt:RequireNewTransaction="false">
<xpdl2:Implementation>
<xpdl2:Task>
<xpdl2:TaskServicexpdExt:ImplementationType="E-Mail"Implementation="Other">
```

```
<xpdl2:MessageInId="_RLGcIFV1EeCITe_kgVEprw"/>
<xpdl2:MessageOutId="_RLGcIVV1EeCITe_kgVEprw"/>
<email:Email>
<email:Definition>
<email:Fromemail:Configuration="Server"/>
<email:To>${ProcessInitiator}</email:To>
<email:Subject>Your Expense Report has been Rejected by your manager :(</email:Subject>
</email:Definition>
<email:Body>Your Expense Report has been Rejected by your manager :(</email:Body>
<email:SMTPemail:Configuration="Server"/>
</email:Email>
</xpdl2:TaskService>
</xpdl2:Task>
</xpdl2:Implementation>
<xpdl2:Performers>
<xpdl2:Performer>_C20eMlV1EeCITe_kgVEprw</xpdl2:Performer>
</xpdl2:Performers>
<xpdl2:Extensions/>
<xpdExt:ActivityResourcePatterns>
<xpdExt:AllocationStrategyxpdExt:Strategy="SYSTEM_DETERMINED"/>
</xpdExt:ActivityResourcePatterns>
</xpdl2:Activity>
<xpdl2:ActivityId="_iBRPYFVyEeCITe_kgVEprw"Name="ApprovalEmail"IsATransaction="false"xpdExt:Visi
bility="Private"xpdExt:DisplayName="Approval Email"xpdExt:RequireNewTransaction="false">
<xpdl2:Implementation>
<xpdl2:Task>
<xpdl2:TaskServicexpdExt:ImplementationType="E-Mail"Implementation="Other">
<xpdl2:MessageInId="_C20eMFV1EeCITe_kgVEprw"/>
<xpdl2:MessageOutId="_C20eMVV1EeCITe_kgVEprw"/>
<email:Email>
<email:Definition>
<email:Fromemail:Configuration="Server"/>
<email:To>${ProcessInitiator}</email:To>
<email:Subject>Your Expense Report has been approved!</email:Subject>
</email:Definition>
<email:Body>Your Expense Report has been approved!</email:Body>
<email:SMTPemail:Configuration="Server"/>
</email:Email>
</xpdl2:TaskService>
</xpdl2:Task>
</xpdl2:Implementation>
<xpdl2:Performers>
<xpdl2:Performer>_C20eMlV1EeCITe_kgVEprw</xpdl2:Performer>
</xpdl2:Performers>
<xpdl2:TransitionRestrictions>
<xpdl2:TransitionRestriction>
<xpdl2:JoinType="Exclusive"ExclusiveType="Data"/>
</xpdl2:TransitionRestriction>
```

```
</xpdl2:TransitionRestrictions>
<xpdl2:Extensions/>
<xpdExt:ActivityResourcePatterns>
<xpdExt:AllocationStrategyxpdExt:Strategy="SYSTEM_DETERMINED"/>
<xpdExt:WorkItemPriorityInitialPriority="50"/>
</xpdExt:ActivityResourcePatterns>
</xpdl2:Activity>
</xpdl2:Activities>
<xpdl2:Transitions>
<xpdl2:TransitionId="_lWzeYFFtEeC7_uBwNhVvUA"xpdExt:DisplayName=""Name=""From="_UMtJ6VFtE
eC7_uBwNhVvUA"To="_iMnaUFFtEeC7_uBwNhVvUA" />
<xpdl2:TransitionId="_FLgmIFFuEeC7_uBwNhVvUA"xpdExt:DisplayName=""Name=""From="_iMnaUFFt
EeC7_uBwNhVvUA"To="_dRqNAVVvEeCITe_kgVEprw" />
<xpdl2:TransitionId="_GZnYAFFuEeC7_uBwNhVvUA"xpdExt:DisplayName="No"Name="No"From="__Q4
awFFtEeC7_uBwNhVvUA"To="_iBRPYFVyEeCITe_kgVEprw">
<xpdl2:ConditionType="OTHERWISE"/>
</xpdl2:Transition>
<xpdl2:TransitionId="_IlUakFFuEeC7_uBwNhVvUA"xpdExt:DisplayName="Yes"Name="Yes"From="__Q4
awFFtEeC7_uBwNhVvUA"To="_tFUpQFFtEeC7_uBwNhVvUA">
<xpdl2:ConditionType="CONDITION">
<xpdl2:ExpressionScriptGrammar="Text">$DATA{Amount} > 500</xpdl2:Expression>
</xpdl2:Condition>
</xpdl2:Transition>
<xpdl2:TransitionId="_fDunkVVvEeCITe_kgVEprw"xpdExt:DisplayName="Yes"Name="Yes"From="_dRqN
AVVvEeCITe_kgVEprw"To="__Q4awFFtEeC7_uBwNhVvUA">
<xpdl2:ConditionType="CONDITION">
<xpdl2:ExpressionScriptGrammar="Text">$DATA{IsManagerApproved} == true</xpdl2:Expression>
</xpdl2:Condition>
</xpdl2:Transition>
<xpdl2:TransitionId="_yy0BUFVvEeCITe_kgVEprw"xpdExt:DisplayName=""Name=""From="_tFUpQFFtEe
C7_uBwNhVvUA"To="_xrOHIFVvEeCITe_kgVEprw" />
<xpdl2:TransitionId="_25_SE1VvEeCITe_kgVEprw"xpdExt:DisplayName="Yes"Name="Yes"From="_xrOHI
FVvEeCITe_kgVEprw"To="_iBRPYFVyEeCITe_kgVEprw">
<xpdl2:ConditionType="CONDITION">
<xpdl2:ExpressionScriptGrammar="Text">$DATA{IsSeniorManagerApproved} ==
true</xpdl2:Expression>
</xpdl2:Condition>
</xpdl2:Transition>
<xpdl2:TransitionId="_9Dj4YFVvEeCITe_kgVEprw"xpdExt:DisplayName="No"Name="No"From="_xrOHIF
VvEeCITe_kgVEprw"To="_wtGAEFVxEeCITe_kgVEprw">
<xpdl2:ConditionType="OTHERWISE"/>
</xpdl2:Transition>
<xpdl2:TransitionId="_ZqKscFVwEeCITe_kgVEprw"xpdExt:DisplayName="No"Name="No"From="_dRqN
AVVvEeCITe_kgVEprw"To="_CyeREVVyEeCITe_kgVEprw">
<xpdl2:ConditionType="OTHERWISE"/>
</xpdl2:Transition>
<xpdl2:TransitionId="_5cqYQFVxEeCITe_kgVEprw"xpdExt:DisplayName=""Name=""From="_wtGAEFVxE
eCITe_kgVEprw"To="_UMtJ6lFtEeC7_uBwNhVvUA" />
```

<xpdl2:TransitionId="_V_k_IFVyEeClTe_kgVEprw"Name=""From="_CyeREVVyEeClTe_kgVEprw"To="_U
MtJ6lFtEeC7_uBwNhVvUA" />
<xpdl2:TransitionId="_zSQPAFVyEeClTe_kgVEprw"Name=""From="_iBRPYFVyEeClTe_kgVEprw"To="_U
MtJ6lFtEeC7_uBwNhVvUA" />
</xpdl2:Transitions>
</xpdl2:WorkflowProcess>
</xpdl2:WorkflowProcesses>
<xpdl2:ExtendedAttributes>
<xpdl2:ExtendedAttributeName="CreatedBy"Value="TIBCO Business Studio"/>
<xpdl2:ExtendedAttributeName="FormatVersion"Value="8"/>
</xpdl2:ExtendedAttributes>
</xpdl2:Package>

Appendix B: Experiment 2 Full Test Results

Class	Method	Hit count	Wallclock milliseconds
WFTester.Program	Main(string[] args)	1	3322.062
OOWF.Lib.Engine.WFEngine	CreateProcessInstance(string definitionName, string initiator, Dictionary<string, string>)	1	989.579
OOWF.Lib.Engine.WFEngine	ValidateDefinitionCacheByName(string defName)	1	882.789
OOWF.Lib.Engine.WFEngine	get_CurrentDefinition()	21	7.038
OOWF.Lib.Entities.Runtime.ProcessDefinition	.ctor()	2	6.741
OOWF.Lib.Entities.Runtime.ProcessDefinition	GetByName(string definitionName)	1	874.862
OOWF.Lib.DAL.ProcessDefinitionDAO	GetByName(string definitionName)	1	843.003
OOWF.Lib.DAL.ProcessDefinitionDAO	CreateFromDataReader(SqlDataReader reader)	1	173.739
OOWF.Lib.Shared.Utils.ProcDefXMLParser	.ctor(string xmlDefinition)	1	38.257
OOWF.Lib.Shared.Utils.ProcDefXMLParser	.ctor()	1	3.496
OOWF.Lib.Shared.Utils.ProcDefXMLParser	Parse()	1	45.465
OOWF.Lib.Shared.Utils.ProcDefXMLParser	ParsePackageHeader()	1	23.949
OOWF.Lib.Shared.Utils.ProcDefXMLParser	ParseParticipants()	1	3.407
OOWF.Lib.Shared.Utils.ProcDefXMLParser	ParseActivities()	1	5.674
OOWF.Lib.Entities.Runtime.ProcessInstance	Execute()	1	70.065
OOWF.Lib.Shared.Utils.Logger	Log(string message, Exception ex)	12	80.009
OOWF.Lib.Entities.Runtime.ProcessInstance	Save()	4	34.051
OOWF.Lib.DAL.ProcessInstanceDAO	Save(ProcessInstance i)	4	30.963
OOWF.Lib.Shared.Utils.DataFieldsParser	GetXML()	4	21.812
OOWF.Lib.Engine.WFEngine	ProcessActivity(Activity a)	8	2346.656
OOWF.Lib.Engine.WFEngine	ProcessEventActivity(EventActivity source)	2	36.539

OOWF.Lib.Entities.Runtime.ActivityInstance	Execute()	2	14.870
OOWF.Lib.Entities.Runtime.ActivityInstance	Save()	3	8.043
OOWF.Lib.DAL.ActivityInstanceDAO	Save(ActivityInstance a)	3	5.766
OOWF.Lib.Engine.WFEngine	Traverse(ActivityInstance completedActivity)	5	2332.248
OOWF.Lib.Engine.WFEngine	ProcessManualTask(ManualTask source)	2	19.566
OOWF.Lib.Engine.WFEngine	ResolveLookupField(string lookup)	8	69.357
OOWF.Lib.Entities.Runtime.User	GetAllForRole(string roleId)	2	3.655
OOWF.Lib.DAL.UserDAO	GetAllForRole(string roleId)	2	2.025
OOWF.Lib.Entities.Runtime.ManualTaskInstance	Execute()	2	11.465
OOWF.Lib.Entities.Runtime.ManualTaskInstance	Save()	4	5.615
OOWF.Lib.Entities.Runtime.ManualTaskInstance	GetByPerformer(string procInstId, StatusEnums, string performer)	2	8.206
OOWF.Lib.DAL.ManualTaskInstanceDAO	GetByPerformer(string procInstId, StatusEnums, string performer)	2	7.936
OOWF.Lib.DAL.ManualTaskInstanceDAO	GetByParams(string id, string procInstId, string name, StatusEnum s, string performer)	2	4.844
OOWF.Lib.DAL.ManualTaskInstanceDAO	CreateFromDataReader(SqlDataReader reader)	2	1.714
OOWF.Lib.Engine.WFEngine	CompleteTask(ManualTaskInstance m)	2	2317.532
OOWF.Lib.Engine.WFEngine	ProcessRouteActivity(RouteActivity source)	3	2314.507
OOWF.Lib.Engine.WFEngine	ParseConditionExpression(string expression)	3	65.874
OOWF.Lib.Shared.Utils.ConditionEvaluator	ResolveDataType(string value)	3	62.891
OOWF.Lib.Shared.Utils.ConditionEvaluator	Evaluate()	3	1.543
OOWF.Lib.Engine.WFEngine	ProcessEmailActivity(EmailTask source)	1	2239.298
OOWF.Lib.Entities.Runtime.EmailTaskInstance	Execute()	1	2230.424
	Totals:	**127**	**21567.537**

Object Oriented System Run #1

Class	Method	Hit count	Wallclock milliseconds
WFTester.Program	Main(string[] args)	1	3297.876
OOWF.Lib.Engine.WFEngine	CreateProcessInstance(string definitionName, string initiator, Dictionary<string, string>)	1	990.078
OOWF.Lib.Engine.WFEngine	ValidateDefinitionCacheByName(string defName)	1	880.851
OOWF.Lib.Engine.WFEngine	get_CurrentDefinition()	21	7.185
OOWF.Lib.Entities.Runtime.ProcessDefinition	.ctor()	2	6.888
OOWF.Lib.Entities.Runtime.ProcessDefinition	GetByName(string definitionName)	1	872.733
OOWF.Lib.DAL.ProcessDefinitionDAO	GetByName(string definitionName)	1	840.186
OOWF.Lib.DAL.ProcessDefinitionDAO	CreateFromDataReader(SqlDataReader reader)	1	171.985
OOWF.Lib.Shared.Utils.ProcDefXMLParser	.ctor(string xmlDefinition)	1	38.421
OOWF.Lib.Shared.Utils.ProcDefXMLParser	.ctor()	1	3.408
OOWF.Lib.Shared.Utils.ProcDefXMLParser	Parse()	1	45.563
OOWF.Lib.Shared.Utils.ProcDefXMLParser	ParsePackageHeader()	1	24.241
OOWF.Lib.Shared.Utils.ProcDefXMLParser	ParseParticipants()	1	3.399
OOWF.Lib.Shared.Utils.ProcDefXMLParser	ParseActivities()	1	5.641
OOWF.Lib.Entities.Runtime.ProcessInstance	Execute()	1	70.059
OOWF.Lib.Shared.Utils.Logger	Log(string message, Exception ex)	12	79.371
OOWF.Lib.Entities.Runtime.ProcessInstance	Save()	4	35.520
OOWF.Lib.DAL.ProcessInstanceDAO	Save(ProcessInstance i)	4	32.420
OOWF.Lib.Shared.Utils.DataFieldsParser	GetXML()	4	22.267
OOWF.Lib.Engine.WFEngine	ProcessActivity(Activity a)	8	2324.866
OOWF.Lib.Engine.WFEngine	ProcessEventActivity(EventActivity source)	2	40.184
OOWF.Lib.Entities.Runtime.ActivityInstance	Execute()	2	12.270
OOWF.Lib.Entities.Runtime.ActivityInstance	Save()	3	6.702

OOWF.Lib.DAL.ActivityInstanceDAO	Save(ActivityInstance a)	3	4.564
OOWF.Lib.Engine.WFEngine	Traverse(ActivityInstance completedActivity)	5	2313.654
OOWF.Lib.Engine.WFEngine	ProcessManualTask(ManualTask source)	2	24.464
OOWF.Lib.Engine.WFEngine	ResolveLookupField(string lookup)	8	70.180
OOWF.Lib.Entities.Runtime.User	GetAllForRole(string roleId)	2	3.937
OOWF.Lib.DAL.UserDAO	GetAllForRole(string roleId)	2	2.182
OOWF.Lib.Entities.Runtime.ManualTaskInstance	Execute()	2	15.865
OOWF.Lib.Entities.Runtime.ManualTaskInstance	Save()	4	6.613
OOWF.Lib.Entities.Runtime.ManualTaskInstance	GetByPerformer(string procInstId, StatusEnums, string performer)	2	7.331
OOWF.Lib.DAL.ManualTaskInstanceDAO	GetByPerformer(string procInstId, StatusEnums, string performer)	2	6.982
OOWF.Lib.DAL.ManualTaskInstanceDAO	GetByParams(string id, string procInstId, string name, StatusEnum s, string performer)	2	4.566
OOWF.Lib.DAL.ManualTaskInstanceDAO	CreateFromDataReader(SqlDataReader reader)	2	1.691
OOWF.Lib.Engine.WFEngine	CompleteTask(ManualTaskInstance m)	2	2292.660
OOWF.Lib.Engine.WFEngine	ProcessRouteActivity(RouteActivity source)	3	2289.839
OOWF.Lib.Engine.WFEngine	ParseConditionExpression(string expression)	3	66.613
OOWF.Lib.Shared.Utils.ConditionEvaluator	ResolveDataType(string value)	3	63.653
OOWF.Lib.Shared.Utils.ConditionEvaluator	Evaluate()	3	1.547
OOWF.Lib.Engine.WFEngine	ProcessEmailActivity(EmailTask source)	1	2215.083
OOWF.Lib.Entities.Runtime.EmailTaskInstance	Execute()	1	2205.308
		Total:	21408.846

Object Oriented System Run #2

Class	Method	Hit count	Wallclock milliseconds
WFTester.Program	Main(string[] args)	1	3308.891
OOWF.Lib.Engine.WFEngine	CreateProcessInstance(string definitionName, string initiator, Dictionary<string, string>)	1	979.768
OOWF.Lib.Engine.WFEngin	ValidateDefinitionCacheByName(string	1	878.679

e	defName)		
OOWF.Lib.Engine.WFEngine	get_CurrentDefinition()	21	7.109
OOWF.Lib.Entities.Runtime.ProcessDefinition	.ctor()	2	6.811
OOWF.Lib.Entities.Runtime.ProcessDefinition	GetByName(string definitionName)	1	870.686
OOWF.Lib.DAL.ProcessDefinitionDAO	GetByName(string definitionName)	1	836.728
OOWF.Lib.DAL.ProcessDefinitionDAO	CreateFromDataReader(SqlDataReader reader)	1	171.154
OOWF.Lib.Shared.Utils.ProcDefXMLParser	.ctor(string xmlDefinition)	1	38.447
OOWF.Lib.Shared.Utils.ProcDefXMLParser	.ctor()	1	3.441
OOWF.Lib.Shared.Utils.ProcDefXMLParser	Parse()	1	45.587
OOWF.Lib.Shared.Utils.ProcDefXMLParser	ParsePackageHeader()	1	24.116
OOWF.Lib.Shared.Utils.ProcDefXMLParser	ParseParticipants()	1	3.477
OOWF.Lib.Shared.Utils.ProcDefXMLParser	ParseActivities()	1	5.463
OOWF.Lib.Engine.WFEngine	UpdateCurrentDataFields(string procInstId, Dictionary<string, string>)	3	1.506
OOWF.Lib.Entities.Runtime.ProcessInstance	Execute()	1	69.649
OOWF.Lib.Shared.Utils.Logger	Log(string message, Exception ex)	12	63.370
OOWF.Lib.Entities.Runtime.ProcessInstance	Save()	4	47.854
OOWF.Lib.DAL.ProcessInstanceDAO	Save(ProcessInstance i)	4	44.735
OOWF.Lib.Shared.Utils.DataFieldsParser	GetXML()	4	22.198
OOWF.Lib.Engine.WFEngine	ProcessActivity(Activity a)	8	2340.205
OOWF.Lib.Engine.WFEngine	ProcessEventActivity(EventActivity source)	2	45.101
OOWF.Lib.Entities.Runtime.ActivityInstance	Execute()	2	12.619
OOWF.Lib.Entities.Runtime.ActivityInstance	Save()	3	7.185
OOWF.Lib.DAL.ActivityInstanceDAO	Save(ActivityInstance a)	3	5.058
OOWF.Lib.Engine.WFEngine	Traverse(ActivityInstance completedActivity)	5	2328.976

OOWF.Lib.Engine.WFEngine	ProcessManualTask(ManualTask source)	2	23.372
OOWF.Lib.Engine.WFEngine	ResolveLookupField(string lookup)	8	70.905
OOWF.Lib.Entities.Runtime.User	GetAllForRole(string roleId)	2	3.582
OOWF.Lib.Entities.Runtime.ManualTaskInstance	Execute()	2	15.868
OOWF.Lib.Entities.Runtime.ManualTaskInstance	Save()	4	12.485
OOWF.Lib.Entities.Runtime.ManualTaskInstance	GetByPerformer(string procInstId, StatusEnums, string performer)	2	6.468
OOWF.Lib.DAL.ManualTaskInstanceDAO	GetByPerformer(string procInstId, StatusEnums, string performer)	2	6.218
OOWF.Lib.DAL.ManualTaskInstanceDAO	GetByParams(string id, string procInstId, string name, StatusEnum s, string performer)	2	4.151
OOWF.Lib.DAL.ManualTaskInstanceDAO	CreateFromDataReader(SqlDataReader reader)	2	1.658
OOWF.Lib.Engine.WFEngine	CompleteTask(ManualTaskInstance m)	2	2316.032
OOWF.Lib.Engine.WFEngine	ProcessRouteActivity(RouteActivity source)	3	2313.298
OOWF.Lib.Engine.WFEngine	ParseConditionExpression(string expression)	3	68.030
OOWF.Lib.Shared.Utils.ConditionEvaluator	ResolveDataType(string value)	3	64.866
OOWF.Lib.Shared.Utils.ConditionEvaluator	Evaluate()	3	1.560
OOWF.Lib.Engine.WFEngine	ProcessEmailActivity(EmailTask source)	1	2230.125
OOWF.Lib.Entities.Runtime.EmailTaskInstance	Execute()	1	2207.458
		Total:	21514.888

Object Oriented System Run #3

	All results in kilobytes				
Step	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Average Memory Usage**
At start	104.664	514.184	513.051	513.051	411.237
After Process Creation	1036.145	895.859	894.727	894.727	930.364
After Manager Approval	1124.145	955.945	954.813	954.813	997.429
After Senior Manager Approval	1432.230	1100.117	1098.984	1098.984	1182.579
				Total Avg:	880.402

Object Oriented System Memory Test

Class	Method	Hit count	Wallclock milliseconds
WFTester.Program	Main(string[] args)	1	3393.816
AOWF.Lib.Engine.WFEngine	CreateProcessInstance(string definitionName, string initiator, Dictionary<string, string>)	1	1062.105
AOWF.Lib.Engine.WFEngine	ValidateDefinitionCacheByName(string defName)	1	880.199
AOWF.Lib.Engine.WFEngine	get_CurrentDefinition()	21	7.296
AOWF.Lib.Entities.Runtime.ProcessDefinition	.ctor()	2	6.998
AOWF.Lib.Entities.Runtime.ProcessDefinition	GetByName(string definitionName)	1	872.038
AOWF.Lib.DAL.ProcessDefinitionDAO	GetByName(string definitionName)	1	839.200
AOWF.Lib.DAL.ProcessDefinitionDAO	CreateFromDataReader(SqlDataReader reader)	1	172.838
AOWF.Lib.Shared.Utils.ProcDefXMLParser	.ctor(string xmlDefinition)	1	39.152
AOWF.Lib.Shared.Utils.ProcDefXMLParser	.ctor()	1	3.796
AOWF.Lib.Shared.Utils.ProcDefXMLParser	Parse()	1	46.051
AOWF.Lib.Shared.Utils.ProcDefXMLParser	ParsePackageHeader()	1	24.660
AOWF.Lib.Shared.Utils.ProcDefXMLParser	ParseParticipants()	1	3.434
AOWF.Lib.Shared.Utils.ProcDefXMLParser	ParseActivities()	1	5.712
AOWF.Lib.Entities.Runtime.ProcessInstance	Execute()	1	71.241
AOWF.Lib.Aspects.ActivityLoggingAspect	OnEntry(MethodExecutionArgs args)	6	42.702
AOWF.Lib.Shared.Utils.Logger	Log(string message, Exception ex)	12	74.336
AOWF.Lib.Entities.Runtime.ProcessInstance	Save()	4	36.116
AOWF.Lib.DAL.ProcessInstanceDAO	Save(ProcessInstance i)	4	33.041
AOWF.Lib.Shared.Utils.DataFieldsParser	GetXML()	4	22.298
AOWF.Lib.Aspects.ActivityLoggingAspect	OnSuccess(MethodExecutionArgs args)	5	7.929
AOWF.Lib.Engine.WFEngine	ProcessActivity(Activity a)	8	2344.013
AOWF.Lib.Engine.WFEngine	ProcessEventActivity(EventActivity source)	2	36.444

AOWF.Lib.Entities.Runtime.ActivityInstance	Execute()	2	13.691
AOWF.Lib.Entities.Runtime.ActivityInstance	Save()	2	7.165
AOWF.Lib.DAL.ActivityInstanceDAO	Save(ActivityInstance a)	2	5.091
AOWF.Lib.Engine.WFEngine	Traverse(ActivityInstance completedActivity)	5	2332.564
AOWF.Lib.Engine.WFEngine	ProcessManualTask(ManualTask source)	2	19.123
AOWF.Lib.Engine.WFEngine	ResolveLookupField(string lookup)	8	70.949
AOWF.Lib.Entities.Runtime.User	GetAllForRole(string roleId)	2	3.701
AOWF.Lib.Entities.Runtime.ManualTaskInstance	Execute()	2	10.070
AOWF.Lib.Entities.Runtime.ManualTaskInstance	Save()	4	5.262
AOWF.Lib.Entities.Runtime.ManualTaskInstance	GetByPerformer(string procInstId, StatusEnums, string performer)	2	7.050
AOWF.Lib.DAL.ManualTaskInstanceDAO	GetByPerformer(string procInstId, StatusEnums, string performer)	2	6.781
AOWF.Lib.DAL.ManualTaskInstanceDAO	GetByParams(string id, string procInstId, string name, StatusEnum s, string performer)	2	4.669
AOWF.Lib.DAL.ManualTaskInstanceDAO	CreateFromDataReader(SqlDataReader reader)	2	1.747
AOWF.Lib.Engine.WFEngine	CompleteTask(ManualTaskInstance m)	2	2317.354
AOWF.Lib.Engine.WFEngine	ProcessRouteActivity(RouteActivity source)	3	2314.460
AOWF.Lib.Engine.WFEngine	ParseConditionExpression(string expression)	3	67.893
AOWF.Lib.Shared.Utils.ConditionEvaluator	ResolveDataType(string value)	3	64.616
AOWF.Lib.Shared.Utils.ConditionEvaluator	Evaluate()	3	1.612
AOWF.Lib.Engine.WFEngine	ProcessEmailActivity(EmailTask source)	1	2237.624
AOWF.Lib.Entities.Runtime.EmailTaskInstance	Execute()	1	2225.452
AOWF.Lib.Aspects.ActivityLoggingAspect	OnException(MethodExecutionArgs args)	1	24.854
		Total:	21767.143

Aspect Oriented System Run #1

Class	Method	Hit count	Wallclock milliseconds
WFTester.Program	Main(string[] args)	1	3375.387
AOWF.Lib.Engine.WFEngine	CreateProcessInstance(string definitionName, string initiator, Dictionary<string, string>)	1	1057.654
AOWF.Lib.Engine.WFEngine	ValidateDefinitionCacheByName(string defName)	1	877.366

AOWF.Lib.Engine.WFEngine	get_CurrentDefinition()	21	7.612
AOWF.Lib.Entities.Runtime.ProcessDefinition	.ctor()	2	7.320
AOWF.Lib.Entities.Runtime.ProcessDefinition	GetByName(string definitionName)	1	868.862
AOWF.Lib.DAL.ProcessDefinitionDAO	GetByName(string definitionName)	1	836.830
AOWF.Lib.DAL.ProcessDefinitionDAO	CreateFromDataReader(SqlDataReader reader)	1	171.763
AOWF.Lib.Shared.Utils.ProcDefXMLParser	.ctor(string xmlDefinition)	1	39.941
AOWF.Lib.Shared.Utils.ProcDefXMLParser	.ctor()	1	3.741
AOWF.Lib.Shared.Utils.ProcDefXMLParser	Parse()	1	45.141
AOWF.Lib.Shared.Utils.ProcDefXMLParser	ParsePackageHeader()	1	24.184
AOWF.Lib.Shared.Utils.ProcDefXMLParser	ParseParticipants()	1	3.376
AOWF.Lib.Shared.Utils.ProcDefXMLParser	ParseActivities()	1	5.617
AOWF.Lib.Entities.Runtime.ProcessInstance	Execute()	1	69.866
AOWF.Lib.Aspects.ActivityLoggingAspect	OnEntry(MethodExecutionArgs args)	6	42.297
AOWF.Lib.Shared.Utils.Logger	Log(string message, Exception ex)	12	73.377
AOWF.Lib.Entities.Runtime.ProcessInstance	Save()	4	35.549
AOWF.Lib.DAL.ProcessInstanceDAO	Save(ProcessInstance i)	4	32.464
AOWF.Lib.Shared.Utils.DataFieldsParser	GetXML()	4	21.960
AOWF.Lib.Aspects.ActivityLoggingAspect	OnSuccess(MethodExecutionArgs args)	5	6.745
AOWF.Lib.Engine.WFEngine	ProcessActivity(Activity a)	8	2330.448
AOWF.Lib.Engine.WFEngine	ProcessEventActivity(EventActivity source)	2	34.543
AOWF.Lib.Entities.Runtime.ActivityInstance	Execute()	2	12.471
AOWF.Lib.Entities.Runtime.ActivityInstance	Save()	2	6.351
AOWF.Lib.DAL.ActivityInstanceDAO	Save(ActivityInstance a)	2	4.249
AOWF.Lib.Engine.WFEngine	Traverse(ActivityInstance completedActivity)	5	2318.237
AOWF.Lib.Engine.WFEngine	ProcessManualTask(ManualTask source)	2	19.335
AOWF.Lib.Engine.WFEngine	ResolveLookupField(string lookup)	8	69.812

AOWF.Lib.Entities.Runtime.User	GetAllForRole(string roleId)	2	3.945
AOWF.Lib.DAL.UserDAO	GetAllForRole(string roleId)	2	2.391
AOWF.Lib.Entities.Runtime.ManualTaskInstance	Execute()	2	10.184
AOWF.Lib.Entities.Runtime.ManualTaskInstance	Save()	4	5.760
AOWF.Lib.Entities.Runtime.ManualTaskInstance	GetByPerformer(string procInstId, StatusEnums, string performer)	2	7.025
AOWF.Lib.DAL.ManualTaskInstanceDAO	GetByPerformer(string procInstId, StatusEnums, string performer)	2	6.756
AOWF.Lib.DAL.ManualTaskInstanceDAO	GetByParams(string id, string procInstId, string name, StatusEnum s, string performer)	2	4.637
AOWF.Lib.DAL.ManualTaskInstanceDAO	CreateFromDataReader(SqlDataReader reader)	2	1.685
AOWF.Lib.Engine.WFEngine	CompleteTask(ManualTaskInstance m)	2	2303.886
AOWF.Lib.Engine.WFEngine	ProcessRouteActivity(RouteActivity source)	3	2301.131
AOWF.Lib.Engine.WFEngine	ParseConditionExpression(string expression)	3	66.498
AOWF.Lib.Shared.Utils.ConditionEvaluator	ResolveDataType(string value)	3	63.483
AOWF.Lib.Shared.Utils.ConditionEvaluator	Evaluate()	3	1.600
AOWF.Lib.Engine.WFEngine	ProcessEmailActivity(EmailTask source)	1	2224.887
AOWF.Lib.Entities.Runtime.EmailTaskInstance	Execute()	1	2214.852
AOWF.Lib.Aspects.ActivityLoggingAspect	OnException(MethodExecutionArgs args)	1	25.502
		Total:	21646.720

Aspect Oriented System Run #2

Class	Method	Hit count	Wallclock milliseconds
WFTester.Program	Main(string[] args)	1	3449.172
AOWF.Lib.Engine.WFEngine	CreateProcessInstance(string definitionName, string initiator, Dictionary<string, string>)	1	1109.162
AOWF.Lib.Engine.WFEngine	ValidateDefinitionCacheByName(string defName)	1	919.985
AOWF.Lib.Engine.WFEngine	get_CurrentDefinition()	21	8.339
AOWF.Lib.Entities.Runtime.ProcessDefinition	.ctor()	2	8.041
AOWF.Lib.Entities.Runtime.ProcessDefinition	GetByName(string definitionName)	1	910.402
AOWF.Lib.DAL.ProcessDefinitionDAO	GetByName(string definitionName)	1	869.834
AOWF.Lib.DAL.ProcessDefi	CreateFromDataReader(SqlDataReader reader)	1	171.514

nitionDAO			
AOWF.Lib.Shared.Utils.ProcDefXMLParser	.ctor(string xmlDefinition)	1	38.964
AOWF.Lib.Shared.Utils.ProcDefXMLParser	.ctor()	1	3.591
AOWF.Lib.Shared.Utils.ProcDefXMLParser	Parse()	1	47.120
AOWF.Lib.Shared.Utils.ProcDefXMLParser	ParsePackageHeader()	1	25.282
AOWF.Lib.Shared.Utils.ProcDefXMLParser	ParseParticipants()	1	3.552
AOWF.Lib.Shared.Utils.ProcDefXMLParser	ParseActivities()	1	5.601
AOWF.Lib.Entities.Runtime.ProcessInstance	Execute()	1	75.035
AOWF.Lib.Aspects.ActivityLoggingAspect	OnEntry(MethodExecutionArgs args)	6	43.688
AOWF.Lib.Shared.Utils.Logger	Log(string message, Exception ex)	12	63.608
AOWF.Lib.Entities.Runtime.ProcessInstance	Save()	4	38.943
AOWF.Lib.DAL.ProcessInstanceDAO	Save(ProcessInstance i)	4	35.897
AOWF.Lib.Shared.Utils.DataFieldsParser	GetXML()	4	25.390
AOWF.Lib.Aspects.ActivityLoggingAspect	OnSuccess(MethodExecutionArgs args)	5	5.424
AOWF.Lib.Engine.WFEngine	ProcessActivity(Activity a)	8	2355.117
AOWF.Lib.Engine.WFEngine	ProcessEventActivity(EventActivity source)	2	36.160
AOWF.Lib.Entities.Runtime.ActivityInstance	Execute()	2	12.493
AOWF.Lib.Entities.Runtime.ActivityInstance	Save()	2	6.952
AOWF.Lib.DAL.ActivityInstanceDAO	Save(ActivityInstance a)	2	4.001
AOWF.Lib.Engine.WFEngine	Traverse(ActivityInstance completedActivity)	5	2341.235
AOWF.Lib.Engine.WFEngine	ProcessManualTask(ManualTask source)	2	19.408
AOWF.Lib.Engine.WFEngine	ResolveLookupField(string lookup)	8	73.842
AOWF.Lib.Entities.Runtime.User	GetAllForRole(string roleId)	2	3.662
AOWF.Lib.DAL.UserDAO	GetAllForRole(string roleId)	2	2.099
AOWF.Lib.Entities.Runtime.ManualTaskInstance	Execute()	2	9.921
AOWF.Lib.Entities.Runtime.ManualTaskInstance	Save()	4	6.394
AOWF.Lib.Entities.Runtime.	GetByPerformer(string procInstId, StatusEnum	2	7.015

ManualTaskInstance	s, string performer)		
AOWF.Lib.DAL.ManualTaskInstanceDAO	GetByPerformer(string procInstId, StatusEnums, string performer)	2	6.772
AOWF.Lib.DAL.ManualTaskInstanceDAO	GetByParams(string id, string procInstId, string name, StatusEnum s, string performer)	2	4.680
AOWF.Lib.DAL.ManualTaskInstanceDAO	CreateFromDataReader(SqlDataReader reader)	2	1.828
AOWF.Lib.Engine.WFEngine	CompleteTask(ManualTaskInstance m)	2	2325.595
AOWF.Lib.Engine.WFEngine	ProcessRouteActivity(RouteActivity source)	3	2322.801
AOWF.Lib.Engine.WFEngine	ParseConditionExpression(string expression)	3	70.977
AOWF.Lib.Shared.Utils.ConditionEvaluator	ResolveDataType(string value)	3	67.698
AOWF.Lib.Shared.Utils.ConditionEvaluator	Evaluate()	3	1.797
AOWF.Lib.Engine.WFEngine	ProcessEmailActivity(EmailTask source)	1	2242.901
AOWF.Lib.Entities.Runtime.EmailTaskInstance	Execute()	1	2234.028
AOWF.Lib.Aspects.ActivityLoggingAspect	OnException(MethodExecutionArgs args)	1	15.707
		Total:	22031.628

Aspect Oriented System Run #3

	All results in kilobytes				
Step	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Average Memory Usage**
At start	104.664	516.082	514.867	514.867	412.620
After Process Creation	1156.145	905.758	904.543	904.543	967.747
After Manager Approval	1244.145	965.844	964.629	964.629	1034.812
After Senior Manager Approval	1472.230	1070.016	1068.801	1068.801	1169.962
				Total Avg:	896.285

Aspect Oriented Memory Test

Appendix C: Stored Procedures

```
-- =============================================
-- Author:                  <Ed Peloquin>
-- Create date: <4-10-2011>
-- Description:   <Gets Activites by params>
-- =============================================
ALTERPROCEDURE [dbo].[ActivityInst_GetByParams]
        @ActivityInstId UNIQUEIDENTIFIER=NULL,
        @ProcInstId    UNIQUEIDENTIFIER=NULL,
        @Name          VARCHAR(50)=NULL,
        @Status        VARCHAR(20)=NULL
AS
BEGIN
        SETNOCOUNTON;
        SELECT*FROM ActivityInst
        WHERE (@ActivityInstId ISNULLOR ActivityInst.ActivityInstId = @ActivityInstId)AND
        (@ProcInstId ISNULLOR ActivityInst.ProcInstId = @ProcInstId)AND
        (@Name ISNULLOR ActivityInst.Name = @Name)AND
        (@Status ISNULLOR ActivityInst.[Status] = @Status);
END

-- =============================================
-- Author:                  <Ed Peloquin>
-- Create date: <4-10-2011>
-- Description:   <Saves an activity instance>
-- =============================================
ALTERPROCEDURE [dbo].[ActivityInst_Save]
        @ActivityInstId UNIQUEIDENTIFIER,
        @ProcInstId    UNIQUEIDENTIFIER,
        @Name          VARCHAR(50),
        @DisplayName   VARCHAR(50)=NULL,
        @CompletedDate  DATETIME=NULL,
        @Status        VARCHAR(20)
AS
BEGIN
        SETNOCOUNTON;
        IFEXISTS(SELECT ActivityInst.ActivityInstId FROM ActivityInst
        WHERE ActivityInst.ActivityInstId = @ActivityInstId)
                BEGIN
                        UPDATE ActivityInst SET ActivityInst.ActivityInstId = @ActivityInstId,
                                        ActivityInst.ProcInstId    = @ProcInstId,
                                        ActivityInst.Name          = @Name,
                                        ActivityInst.DisplayName   = @DisplayName,
                                        ActivityInst.CompletedDate  = @CompletedDate,
                                        ActivityInst.[Status]      = @Status
```

```
                WHERE ActivityInst.ActivityInstId = @ActivityInstId;
        END
    ELSE
        BEGIN
            INSERTINTO ActivityInst(ActivityInst.ActivityInstId, ActivityInst.ProcInstId,
                    ActivityInst.Name, ActivityInst.DisplayName,
                    ActivityInst.CreatedDate, ActivityInst.CompletedDate,
                    ActivityInst.[Status])
            VALUES      (@ActivityInstId,@ProcInstId,@Name,@DisplayName,
            GETDATE(),@CompletedDate,@Status);
        END
END

-- =============================================
-- Author:          <Ed Peloquin>
-- Create date: <4-16-2011>
-- Description:   <Gets Manual Tasks by parameters>
-- =============================================
ALTERPROCEDURE [dbo].[ManualTask_GetByParams]
    @ActivityInstId UNIQUEIDENTIFIER=NULL,
    @ProcInstId    UNIQUEIDENTIFIER=NULL,
    @Name         VARCHAR(50)=NULL,
    @Status       VARCHAR(20)=NULL,
    @Performer    VARCHAR(50)=NULL
AS
BEGIN
    SETNOCOUNTON;
    SELECT ActivityInst.*, ManualTask.Performer FROM ActivityInst INNERJOIN ManualTask
    ON ActivityInst.ActivityInstId = ManualTask.ManualTaskId
    WHERE (@ActivityInstId ISNULLOR ActivityInst.ActivityInstId = @ActivityInstId)AND
    (@ProcInstId ISNULLOR ActivityInst.ProcInstId = @ProcInstId)AND
    (@Name ISNULLOR ActivityInst.Name = @Name)AND
    (@Status ISNULLOR ActivityInst.[Status] = @Status)AND
    (@Performer ISNULLOR ManualTask.Performer = @Performer);
END

-- =============================================
-- Author:          <Ed Peloquin>
-- Create date: <4-16-2011>
-- Description:   <Saves a manual task>
-- =============================================
ALTERPROCEDURE [dbo].[ManualTask_Save]
    @ActivityInstId UNIQUEIDENTIFIER,
    @ProcInstId    UNIQUEIDENTIFIER,
    @Name         VARCHAR(50),
    @DisplayName    VARCHAR(50)=NULL,
    @CompletedDate  DATETIME=NULL,
    @Status       VARCHAR(20),
```

```
	@Performer	VARCHAR(50)
AS
BEGIN
	SETNOCOUNTON;

	BEGINTRANSACTION

	-- Save the base activity instance
	EXEC
ActivityInst_Save@ActivityInstId,@ProcInstId,@Name,@DisplayName,@CompletedDate,@Status;

	IF@@ERROR<> 0
		BEGIN
			-- Rollback the transaction
			ROLLBACKTRANSACTION

			-- Raise an error and return
			RAISERROR ('Error in saving an activity instance', 16, 1)
			RETURN
		END

	-- Save the manual task
	IFEXISTS(SELECT ManualTask.ManualTaskId FROM ManualTask
	WHERE ManualTask.ManualTaskId = @ActivityInstId)
		BEGIN
			UPDATE ManualTask SET ManualTask.Performer = @Performer
			WHERE ManualTask.ManualTaskId = @ActivityInstId;
		END
	ELSE
		BEGIN
			INSERTINTO ManualTask(ManualTask.ManualTaskId, ManualTask.Performer)
			VALUES		(@ActivityInstId, @Performer);
		END

	IF@@ERROR<> 0
		BEGIN
			-- Rollback the transaction
			ROLLBACK

			-- Raise an error and return
			RAISERROR ('Error in saving a manual task', 16, 1)
			RETURN
		END

	COMMITTRANSACTION
END

-- ============================================
```

```
-- Author:              <Ed Peloquin>
-- Create date: <4-16-2011>
-- Description:   <Gets Process Definition By Id>
-- =============================================
ALTERPROCEDURE [dbo].[ProcDef_GetById]
        @ProcDefId UNIQUEIDENTIFIER
AS
BEGIN
        SETNOCOUNTON;
        SELECT*FROM ProcDef
        WHERE ProcDef.ProcDefId = @ProcDefId;
END

-- =============================================
-- Author:              <Ed Peloquin>
-- Create date: <4-8-2011>
-- Description:   <Gets the most current process definition by name>
-- =============================================
ALTERPROCEDURE [dbo].[ProcDef_GetMostCurrentByName]
        -- Add the parameters for the stored procedure here
        @Name VARCHAR(100)
AS
BEGIN
        SETNOCOUNTON;
        SELECT*FROM ProcDef where ProcDef.[Name] = @Name and ProcDef.[Version] =
        (SELECTMAX(ProcDef.[Version])FROM ProcDef WHERE ProcDef.[Name] = @Name);
END

-- =============================================
-- Author:              <Ed Peloquin>
-- Create date: <4-8-2011>
-- Description:   <Saves a Process Definition>
-- =============================================
ALTERPROCEDURE [dbo].[ProcDef_Save]
        -- Add the parameters for the stored procedure here
        @ProcDefId   UNIQUEIDENTIFIER,
        @Author      VARCHAR(50),
        @Description VARCHAR(MAX)=NULL,
        @Name        VARCHAR(100),
        @DisplayName VARCHAR(100)=NULL,
        @Status      VARCHAR(20),
        @Version     VARCHAR(10),
        @Definition  TEXT
AS
BEGIN
        SETNOCOUNTON;

        IFEXISTS(SELECT ProcDef.ProcDefId FROM ProcDef WHERE ProcDef.ProcDefId = @ProcDefId)
```

```
                BEGIN
                        -- Update
                        UPDATE ProcDef SET ProcDef.[Author]     = @Author,
                                                                ProcDef.[Description] = @Description,
                                                                ProcDef.[Name]       = @Name,
                                                                ProcDef.[DisplayName] = @DisplayName,
                                                                ProcDef.[Status]     = @Status,
                                                                ProcDef.[Version]    = @Version,
                                                                ProcDef.[Definition]  = @Definition
                        WHERE ProcDef.ProcDefId = @ProcDefId;
                END
        ELSE
                BEGIN
                -- Insert
                INSERTINTO ProcDef(ProcDef.[Author],ProcDef.[CreatedDate],
                        ProcDef.[Definition],ProcDef.[Description],
                        ProcDef.[ProcDefId],ProcDef.[Name],ProcDef.[Status],
                        ProcDef.[Version],ProcDef.[DisplayName])
                VALUES          (@Author,GETDATE(),@Definition,@Description,
                        @ProcDefId,@Name,@Status,@Version,@DisplayName);
                END
END

-- =============================================
-- Author:                <Ed Peloquin>
-- Create date: <4-17-2011>
-- Description:   <Gets Process Instances By Params>
-- =============================================
ALTERPROCEDURE [dbo].[ProcInst_GetByParams]
        @ProcInstId UNIQUEIDENTIFIER=NULL,
        @ProcDefId  UNIQUEIDENTIFIER=NULL,
        @Initiator  VARCHAR(50)=NULL,
        @Status     VARCHAR(20)=NULL,
        @Name       VARCHAR(250)=NULL
AS
BEGIN
        SETNOCOUNTON;

SELECT*FROM ProcInst
WHERE (@ProcInstId ISNULLOR ProcInst.ProcInstId = @ProcInstId)AND
(@ProcDefId ISNULLOR ProcInst.ProcDefId = @ProcDefId)AND
(@Initiator ISNULLOR ProcInst.[Initiator] = @Initiator)AND
(@Status ISNULLOR ProcInst.[Status] = @Status)AND
(@Name ISNULLOR ProcInst.Name = @Name);
END

-- =============================================
-- Author:                <Ed Peloquin>
```

```
-- Create date: <4-16-2011>
-- Description:   <Saves a process instance>
-- =============================================
ALTERPROCEDURE [dbo].[ProcInst_Save]
        @ProcInstId    UNIQUEIDENTIFIER,
        @ProcDefId     UNIQUEIDENTIFIER,
        @Initiator     VARCHAR(50),
        @Name          VARCHAR(250),
        @Status        VARCHAR(20),
        @CompletedDate DATETIME=NULL,
        @XML           TEXT=NULL
AS
BEGIN
        SETNOCOUNTON;
        BEGINTRANSACTION

        IFEXISTS(SELECT ProcInst.ProcInstId FROM ProcInst
        WHERE ProcInst.ProcInstId = @ProcInstId)
                BEGIN
                        UPDATE ProcInst SET ProcInst.ProcInstId    = @ProcInstId,
                                        ProcInst.ProcDefId     = @ProcDefId,
                                        ProcInst.[Initiator]   = @Initiator,
                                        ProcInst.Name          = @Name,
                                        ProcInst.CompletedDate = @CompletedDate,
                                        ProcInst.[Status]      = @Status
                        WHERE ProcInst.ProcInstId = @ProcInstId;
                END
        ELSE
                BEGIN
                        INSERTINTO ProcInst(ProcInst.ProcInstId, ProcInst.ProcDefId,
                                        ProcInst.[Initiator], ProcInst.[Status],
                                        ProcInst.Name, ProcInst.CreatedDate,
                                        ProcInst.CompletedDate)
                        VALUES          (@ProcInstId,@ProcDefId,@Initiator,@Status,
                                        @Name,GETDATE(),@CompletedDate);
                END

        IF@@ERROR<> 0
                BEGIN
                        -- Rollback the transaction
                        ROLLBACKTRANSACTION

                        -- Raise an error and return
                        RAISERROR ('Error saving a process instance', 16, 1)
                        RETURN
                END

        EXEC ProcInstData_Save@ProcInstId,@XML;
```

```
        IF@@ERROR<> 0
                BEGIN
                        -- Rollback the transaction
                        ROLLBACKTRANSACTION

                        -- Raise an error and return
                        RAISERROR ('Error in saving a process instance data fields', 16, 1)
                        RETURN
                END

        COMMITTRANSACTION
END

-- =============================================
-- Author:                <Ed Peloquin>
-- Create date: <4-16-2011>
-- Description:   <Gets Process Instance Data by process instance id>
-- =============================================
ALTERPROCEDURE [dbo].[ProcInstData_GetByProcInstId]
        @ProcInstId UNIQUEIDENTIFIER
AS
BEGIN
        SETNOCOUNTON;

        SELECT*FROM ProcInstData
        WHERE ProcInstData.ProcInstId = @ProcInstId;
END

-- =============================================
-- Author:                <Ed Peloquin>
-- Create date: <4-16-2011>
-- Description:   <Saves Process Instance Data>
-- =============================================
ALTERPROCEDURE [dbo].[ProcInstData_Save]
        @ProcInstId UNIQUEIDENTIFIER,
        @XML      TEXT=NULL
AS
BEGIN
        SETNOCOUNTON;
        IFEXISTS(SELECT ProcInstData.ProcInstId FROM ProcInstData
        WHERE ProcInstData.ProcInstId = @ProcInstId)
                BEGIN
                        UPDATE ProcInstData SET ProcInstData.[XML] = @XML
                        WHERE ProcInstData.ProcInstId = @ProcInstId;
                END
        ELSE
                BEGIN
```

```
                INSERTINTO ProcInstData(ProcInstData.ProcInstId, ProcInstData.[XML])
                VALUES          (@ProcInstId,@XML);
        END
END

-- =============================================
-- Author:              <Ed Peloquin>
-- Create date: <4-16-2011>
-- Description:   <Gets all roles for a user>
-- =============================================
ALTERPROCEDURE [dbo].[Role_GetAllForUser]
        @UserId VARCHAR(50)
AS
BEGIN
        SETNOCOUNTON;

        SELECT [Role].*FROM [Role] INNERJOIN RoleMember
        ON [Role].RoleId = RoleMember.RoleId
        WHERE RoleMember.UserId = @UserId;
END

-- =============================================
-- Author:              <Ed Peloquin>
-- Create date: <4-16-2011>
-- Description:   <Gets a role by Id>
-- =============================================
ALTERPROCEDURE [dbo].[Role_GetById]
        @RoleId VARCHAR(50)
AS
BEGIN
        SETNOCOUNTON;

SELECT*FROM [Role]
WHERE [Role].RoleId = @RoleId;
END

-- =============================================
-- Author:              <Ed Peloquin>
-- Create date: <4-16-2011>
-- Description:   <Saves a Role>
-- =============================================
ALTERPROCEDURE [dbo].[Role_Save]
        @RoleId     VARCHAR(50),
        @DisplayName VARCHAR(50)
AS
BEGIN
        SETNOCOUNTON;
        IFEXISTS(SELECT [Role].RoleId FROM [Role]
```

```
        WHERE [Role].RoleId = @RoleId)
                BEGIN
                        UPDATE [Role] SET [Role].DisplayName = @DisplayName
                        WHERE [Role].RoleId = @RoleId;
                END
        ELSE
                BEGIN
                        INSERTINTO [Role]([Role].RoleId, [Role].DisplayName)
                        VALUES          (@RoleId,@DisplayName);
                END
END

-- =============================================
-- Author:                <Ed Peloquin>
-- Create date: <4-17-2011>
-- Description:   <Gets all users in a role>
-- =============================================
ALTERPROCEDURE [dbo].[User_GetAllForRole]
        @RoleId VARCHAR(50)
AS
BEGIN
        SETNOCOUNTON;

SELECT [User].*FROM [User] INNERJOIN RoleMember
ON [User].UserId = RoleMember.UserId
WHERE RoleMember.RoleId = @RoleId
ORDERBY [User].UserId;
END

-- =============================================
-- Author:                <Ed Peloquin>
-- Create date: <4-8-2011>
-- Description:   <Retrieves a user record by id>
-- =============================================
ALTERPROCEDURE [dbo].[User_GetById]
        @UserId VARCHAR(50)
AS
BEGIN
        SETNOCOUNTON;
        SELECT*FROM [User] WHERE [User].[UserId] = @UserId;
END

-- =============================================
-- Author:                <Ed Peloquin>
-- Create date: <4-8-2011>
-- Description:   <Saves a User record>
-- =============================================
ALTERPROCEDURE [dbo].[User_Save]
```

```
        -- Add the parameters for the stored procedure here
        @UserId     VARCHAR(50),
        @DisplayName VARCHAR(50)=NULL,
        @Email      VARCHAR(50)=NULL
AS
BEGIN
        SETNOCOUNTON;

        IFEXISTS(SELECT [User].UserId FROM [User] WHERE [User].UserId = @UserId)
                BEGIN
                        -- Update
                        UPDATE [User] SET [User].[DisplayName] = @DisplayName,
                                                                [User].[Email]      = @Email
                        WHERE [User].[UserId] = @UserId;
                END
        ELSE
                BEGIN
                -- Insert
                INSERTINTO [User]([User].[UserId],[User].[DisplayName],[User].[Email])
                VALUES          (@UserId,@DisplayName,@Email);
                END
END
```

Appendix D: Object Oriented Source Code

```
namespace OOWF.Lib.DAL.Base
{
///<summary>
/// Base data access object class
///</summary>
publicabstractclassDAOBase
  {
    #region Protected Properties

///<summary>
/// The database connection string
///</summary>
protectedstring ConnectionString
    {
get
      {
return@"Data Source=peloquinej-7lt\sqlexpress;Initial Catalog=WFDB;Integrated Security=True";
      }
    }

    #endregion
  }
}


using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using OOWF.Lib.DAL.Base;
using OOWF.Lib.Entities.Runtime;
using OOWF.Lib.Shared.Enums;
using OOWF.Lib.Shared.Utils;

namespace OOWF.Lib.DAL
{
///<summary>
/// Retrieves and Persists Activities
///</summary>
publicclassActivityInstanceDAO : DAOBase
  {
    #region Constructors

///<summary>
```

```
/// Default Constructor
///</summary>
public ActivityInstanceDAO() { }

      #endregion

      #region Public Methods

///<summary>
/// Gets Activity Instances by Id
///</summary>
///<param name="id"></param>
///<returns></returns>
publicActivityInstance GetById(string id)
      {
List<ActivityInstance> lst = this.GetByParams(id, string.Empty, string.Empty, StatusEnum.NONE);
if (lst[0] != null)
         {
return lst[0];
         }
returnnull;
      }

///<summary>
/// Gets Activity Instances by Name
///</summary>
///<param name="name"></param>
///<returns></returns>
publicActivityInstance GetByName(string name)
      {
List<ActivityInstance> lst = this.GetByParams(string.Empty, string.Empty, name, StatusEnum.NONE);
if (lst[0] != null)
         {
return lst[0];
         }
returnnull;
      }

///<summary>
/// Gets Activity Instances by Process instance id
///</summary>
///<param name="procDefId"></param>
///<returns></returns>
publicList<ActivityInstance> GetByProcInstId(string procInstId)
      {
returnthis.GetByParams(string.Empty, procInstId, string.Empty, StatusEnum.NONE);
      }
```

```
///<summary>
/// Gets Activity Instances by Process instance id and status
///</summary>
///<param name="procDefId"></param>
///<param name="s"></param>
///<returns></returns>
publicList<ActivityInstance> GetByProcInstId(string procDefId, StatusEnum s)
    {
returnthis.GetByParams(string.Empty, procDefId, string.Empty, s);
    }

///<summary>
/// Saves the activity instance to the database
///</summary>
///<param name="a"></param>
publicvoid Save(ActivityInstance a)
    {
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
        {
            conn.Open();
using (SqlCommand cmd = newSqlCommand("ActivityInst_Save", conn))
            {
                cmd.CommandType = System.Data.CommandType.StoredProcedure;
                cmd.Parameters.Add(newSqlParameter("@ActivityInstId", a.Id));
                cmd.Parameters.Add(newSqlParameter("@ProcInstId", a.ProcessInstanceId));
                cmd.Parameters.Add(newSqlParameter("@Name", a.Name));
                cmd.Parameters.Add(newSqlParameter("@DisplayName", a.DisplayName));
if (a.CompletedDate.HasValue)
                {
                    cmd.Parameters.Add(newSqlParameter("@CompletedDate", a.CompletedDate));
                }
                cmd.Parameters.Add(newSqlParameter("@Status", a.Status.ToString()));
                cmd.ExecuteNonQuery();
            }
        }
    }

    #endregion

    #region Private Methods

///<summary>
/// Creates the process definition from the sql data reader that is
/// returned from the query
///</summary>
///<param name="reader"></param>
///<returns></returns>
privateActivityInstance CreateFromDataReader(SqlDataReader reader)
```

```
        {
ActivityInstance a = newActivityInstance();
            a.Id = reader.IsDBNull(reader.GetOrdinal("ActivityInstId")) ? string.Empty :
((Guid)reader["ActivityInstId"]).ToString();
            a.Name = reader.IsDBNull(reader.GetOrdinal("Name")) ? string.Empty : (string)reader["Name"];
            a.CreatedDate = reader.IsDBNull(reader.GetOrdinal("CreatedDate")) ? DateTime.MinValue :
(DateTime)reader["CreatedDate"];
            a.DisplayName = reader.IsDBNull(reader.GetOrdinal("DisplayName")) ? string.Empty :
(string)reader["DisplayName"];
            a.CompletedDate = reader.IsDBNull(reader.GetOrdinal("CompletedDate")) ? DateTime.MinValue :
(DateTime)reader["CompletedDate"];
            a.ProcessInstanceId = reader.IsDBNull(reader.GetOrdinal("ProcInstId")) ? string.Empty :
(string)reader["ProcInstId"];
            a.Status = reader.IsDBNull(reader.GetOrdinal("Status")) ? StatusEnum.NONE :
EnumTranslator.GetStatus((string)reader["Status"]);
return a;
        }

///<summary>
/// Queries for activity instances
///</summary>
///<param name="id"></param>
///<param name="procInstId"></param>
///<param name="name"></param>
///<param name="s"></param>
///<returns></returns>
privateList<ActivityInstance> GetByParams(string id, string procInstId, string name, StatusEnum s)
        {
List<ActivityInstance> lst = newList<ActivityInstance>();
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
            {
                conn.Open();
using (SqlCommand cmd = newSqlCommand("ActivityInst_GetByParams", conn))
                {
if (!string.IsNullOrEmpty(id))
                    {
                        cmd.Parameters.Add(newSqlParameter("@ActivityInstId", id));
                    }
if (!string.IsNullOrEmpty(procInstId))
                    {
                        cmd.Parameters.Add(newSqlParameter("@ProcInstId", procInstId));
                    }
if (!string.IsNullOrEmpty(name))
                    {
                        cmd.Parameters.Add(newSqlParameter("@Name", name));
                    }
if (s != StatusEnum.NONE)
                    {
```

```
                cmd.Parameters.Add(newSqlParameter("@Status", s.ToString()));
            }
            cmd.CommandType = System.Data.CommandType.StoredProcedure;
using (SqlDataReader reader = cmd.ExecuteReader())
            {
if (reader.HasRows)
                {
while (reader.Read())
                    {
                        lst.Add(CreateFromDataReader(reader));
                    }
                }
            }
        }
    }

return lst;
    }

    #endregion
  }
}


using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using OOWF.Lib.DAL.Base;
using OOWF.Lib.Entities.Runtime;
using OOWF.Lib.Shared.Enums;
using OOWF.Lib.Shared.Utils;

namespace OOWF.Lib.DAL
{
///<summary>
/// Retrieves and Persists Manual Tasks
///</summary>
publicclassManualTaskInstanceDAO : DAOBase
  {
    #region Constructors

///<summary>
/// Default Constructor
///</summary>
public ManualTaskInstanceDAO() { }

    #endregion
```

```
        #region Public Methods

///<summary>
/// Gets Manual Tasks by Id
///</summary>
///<param name="id"></param>
///<returns></returns>
publicManualTaskInstance GetById(string id)
        {
List<ManualTaskInstance> lst = this.GetByParams(id, string.Empty, string.Empty, StatusEnum.NONE,
string.Empty);
if (lst[0] != null)
            {
return lst[0];
            }
returnnull;
        }

///<summary>
/// Gets Manual Tasks by Name
///</summary>
///<param name="procInstId"></param>
///<param name="name"></param>
///<returns></returns>
publicManualTaskInstance GetByName(string procInstId, string name)
        {
List<ManualTaskInstance> lst = this.GetByParams(string.Empty, string.Empty, name, StatusEnum.NONE,
string.Empty);
if (lst[0] != null)
            {
return lst[0];
            }
returnnull;
        }

///<summary>
/// Gets Manual Tasks by Process instance id
///</summary>
///<param name="procInstId"></param>
///<returns></returns>
publicList<ManualTaskInstance> GetByProcInstId(string procInstId)
        {
returnthis.GetByParams(string.Empty, procInstId, string.Empty, StatusEnum.NONE, string.Empty);
        }

///<summary>
/// Gets Manual Tasks by Process instance id and status
///</summary>
```

```
///<param name="procDefId"></param>
///<param name="s"></param>
///<returns></returns>
publicList<ManualTaskInstance> GetByProcInstId(string procInstId, StatusEnum s)
      {
returnthis.GetByParams(string.Empty, procInstId, string.Empty, s, string.Empty);
      }

///<summary>
/// Gets Manual Tasks by Process instance id, status, and performer
///</summary>
///<param name="procInstId"></param>
///<param name="s"></param>
///<param name="performer"></param>
///<returns></returns>
publicList<ManualTaskInstance> GetByPerformer(string procInstId, StatusEnum s, string performer)
      {
returnthis.GetByParams(string.Empty, procInstId, string.Empty, s, performer);
      }

///<summary>
/// Saves the activity instance to the database
///</summary>
///<param name="a"></param>
publicvoid Save(ManualTaskInstance m)
      {
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
         {
            conn.Open();
using (SqlCommand cmd = newSqlCommand("ManualTask_Save", conn))
            {
               cmd.CommandType = System.Data.CommandType.StoredProcedure;
               cmd.Parameters.Add(newSqlParameter("@ActivityInstId", m.Id));
               cmd.Parameters.Add(newSqlParameter("@ProcInstId", m.ProcessInstanceId));
               cmd.Parameters.Add(newSqlParameter("@Name", m.Name));
               cmd.Parameters.Add(newSqlParameter("@DisplayName", m.DisplayName));
if (m.CompletedDate.HasValue)
               {
                  cmd.Parameters.Add(newSqlParameter("@CompletedDate", m.CompletedDate));
               }
               cmd.Parameters.Add(newSqlParameter("@Status", m.Status.ToString()));
               cmd.Parameters.Add(newSqlParameter("@Performer", m.Performer.Id));
               cmd.ExecuteNonQuery();
            }
         }
      }

      #endregion
```

```
        #region Private Methods

///<summary>
/// Creates the process definition from the sql data reader that is
/// returned from the query
///</summary>
///<param name="reader"></param>
///<returns></returns>
privateManualTaskInstance CreateFromDataReader(SqlDataReader reader)
        {
ManualTaskInstance m = newManualTaskInstance();
            m.Id = reader.IsDBNull(reader.GetOrdinal("ActivityInstId")) ? string.Empty :
((Guid)reader["ActivityInstId"]).ToString();
            m.Name = reader.IsDBNull(reader.GetOrdinal("Name")) ? string.Empty : (string)reader["Name"];
            m.CreatedDate = reader.IsDBNull(reader.GetOrdinal("CreatedDate")) ? DateTime.MinValue :
(DateTime)reader["CreatedDate"];
            m.DisplayName = reader.IsDBNull(reader.GetOrdinal("DisplayName")) ? string.Empty :
(string)reader["DisplayName"];
            m.CompletedDate = reader.IsDBNull(reader.GetOrdinal("CompletedDate")) ? DateTime.MinValue
: (DateTime)reader["CompletedDate"];
            m.ProcessInstanceId = reader.IsDBNull(reader.GetOrdinal("ProcInstId")) ? string.Empty :
((Guid)reader["ProcInstId"]).ToString();
            m.Status = reader.IsDBNull(reader.GetOrdinal("Status")) ? StatusEnum.NONE :
EnumTranslator.GetStatus((string)reader["Status"]);
            m.Performer.Id = reader.IsDBNull(reader.GetOrdinal("Performer")) ? string.Empty :
(string)reader["Performer"];
return m;
        }

///<summary>
/// Queries for activity instances
///</summary>
///<param name="id"></param>
///<param name="procInstId"></param>
///<param name="name"></param>
///<param name="s"></param>
///<returns></returns>
privateList<ManualTaskInstance> GetByParams(string id, string procInstId, string name, StatusEnum s,
string performer)
        {
List<ManualTaskInstance> lst = newList<ManualTaskInstance>();
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
            {
                conn.Open();
using (SqlCommand cmd = newSqlCommand("ManualTask_GetByParams", conn))
                {
if (!string.IsNullOrEmpty(id))
```

```
                {
                    cmd.Parameters.Add(newSqlParameter("@ActivityInstId", id));
                }
if (!string.IsNullOrEmpty(procInstId))
                {
                    cmd.Parameters.Add(newSqlParameter("@ProcInstId", procInstId));
                }
if (!string.IsNullOrEmpty(name))
                {
                    cmd.Parameters.Add(newSqlParameter("@Name", name));
                }
if (s != StatusEnum.NONE)
                {
                    cmd.Parameters.Add(newSqlParameter("@Status", s.ToString()));
                }
if (!string.IsNullOrEmpty(performer))
                {
                    cmd.Parameters.Add(newSqlParameter("@Performer", performer));
                }
                cmd.CommandType = System.Data.CommandType.StoredProcedure;
using (SqlDataReader reader = cmd.ExecuteReader())
                {
if (reader.HasRows)
                    {
while (reader.Read())
                        {
                            lst.Add(CreateFromDataReader(reader));
                        }
                    }
                }
            }
        }

return lst;
      }

      #endregion
   }
}


using System;
using System.Data.SqlClient;
using OOWF.Lib.DAL.Base;
using OOWF.Lib.Entities.Runtime;
using OOWF.Lib.Shared.Utils;
using OOWF.Lib.Engine;
```

```
namespace OOWF.Lib.DAL
{
///<summary>
/// Instances of this class are responsible for saving and loading
/// process definitions from the database
///</summary>
publicclassProcessDefinitionDAO : DAOBase
  {
    #region Constructors

///<summary>
/// Default Constructor
///</summary>
///<param name="connectionString"></param>
public ProcessDefinitionDAO() { }

    #endregion

    #region Public Methods

///<summary>
/// Retrieves a process definition by it's id
///</summary>
///<param name="id"></param>
///<returns></returns>
publicProcessDefinition GetById(string id)
    {
ProcessDefinition p = null;
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
       {
          conn.Open();
using (SqlCommand cmd = newSqlCommand("ProcDef_GetById", conn))
          {
             cmd.Parameters.Add(newSqlParameter("@ProcDefId", id));
             cmd.CommandType = System.Data.CommandType.StoredProcedure;
using (SqlDataReader reader = cmd.ExecuteReader())
             {
if (reader.HasRows)
                {
                   reader.Read();
                   p = CreateFromDataReader(reader);
                }
             }
          }
       }
return p;
    }
```

```
///<summary>
/// Retrieves a process by it's name
///</summary>
///<param name="processName"></param>
///<returns></returns>
publicProcessDefinition GetByName(string definitionName)
     {
ProcessDefinition p = null;
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
        {
           conn.Open();
using (SqlCommand cmd = newSqlCommand("ProcDef_GetMostCurrentByName", conn))
           {
              cmd.Parameters.Add(newSqlParameter("@Name", definitionName));
              cmd.CommandType = System.Data.CommandType.StoredProcedure;
using (SqlDataReader reader = cmd.ExecuteReader())
              {
if (reader.HasRows)
                 {
                    reader.Read();
                    p = CreateFromDataReader(reader);
                 }
              }
           }
        }
return p;
     }

///<summary>
/// Saves a process definition to the database
///</summary>
///<param name="p"></param>
publicvoid Save(ProcessDefinition p)
     {
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
        {
           conn.Open();
using (SqlCommand cmd = newSqlCommand("ProcDef_Save", conn))
           {
              cmd.CommandType = System.Data.CommandType.StoredProcedure;
              cmd.Parameters.Add(newSqlParameter("@ProcDefId", p.Id));
              cmd.Parameters.Add(newSqlParameter("@Author", p.Author));
              cmd.Parameters.Add(newSqlParameter("@Description", p.Description));
              cmd.Parameters.Add(newSqlParameter("@Name", p.Name));
              cmd.Parameters.Add(newSqlParameter("@DisplayName", p.DisplayName));
              cmd.Parameters.Add(newSqlParameter("@Status", p.Status.ToString()));
              cmd.Parameters.Add(newSqlParameter("@Version", p.Version));
              cmd.Parameters.Add(newSqlParameter("@Definition", p.XMLDefinition));
```

```
                cmd.ExecuteNonQuery();
            }
        }
    }

    #endregion

    #region Private Methods

///<summary>
/// Creates the process definition from the sql data reader that is
/// returned from the query
///</summary>
///<param name="reader"></param>
///<returns></returns>
privatestaticProcessDefinition CreateFromDataReader(SqlDataReader reader)
    {
ProcessDefinition p = newProcessDefinition();
        p.Id = reader.IsDBNull(reader.GetOrdinal("ProcDefId")) ? string.Empty :
((Guid)reader["ProcDefId"]).ToString();
        p.Author = reader.IsDBNull(reader.GetOrdinal("Author")) ? string.Empty :
(string)reader["Author"];
        p.CreatedDate = reader.IsDBNull(reader.GetOrdinal("CreatedDate")) ? DateTime.MinValue :
(DateTime)reader["CreatedDate"];
        p.Description = reader.IsDBNull(reader.GetOrdinal("Description")) ? string.Empty :
(string)reader["Description"];
        p.Name = reader.IsDBNull(reader.GetOrdinal("Name")) ? string.Empty : (string)reader["Name"];
        p.DisplayName = reader.IsDBNull(reader.GetOrdinal("DisplayName")) ? string.Empty :
(string)reader["DisplayName"];
//p.Status = reader.IsDBNull(reader.GetOrdinal("Status")) ? StatusEnum.ACTIVE :
(StatusEnum)reader["DisplayName"];
        p.Version = reader.IsDBNull(reader.GetOrdinal("Version")) ? string.Empty :
(string)reader["Version"];
        p.XMLDefinition = p.DisplayName = reader.IsDBNull(reader.GetOrdinal("Definition")) ?
string.Empty : (string)reader["Definition"];

ProcDefXMLParser parser = newProcDefXMLParser(p.XMLDefinition);
        p.Process = parser.Parse();

return p;
    }

    #endregion
  }
}


using System;
```

```
using System.Collections.Generic;
using System.Data.SqlClient;
using OOWF.Lib.DAL.Base;
using OOWF.Lib.Engine;
using OOWF.Lib.Entities.Runtime;
using OOWF.Lib.Shared.Enums;
using OOWF.Lib.Shared.Utils;

namespace OOWF.Lib.DAL
{
///<summary>
/// Saves and retrieves process instance records
///</summary>
publicclassProcessInstanceDAO : DAOBase
  {
    #region Instance Data

privateDataFieldsParser _dfParser;

privateProcessInstanceDataDAO _dataDAO;

    #endregion

    #region Private Properties

///<summary>
/// This objects custom attributes data access object
///</summary>
privateProcessInstanceDataDAO DataDAO
    {
get
      {
if (_dataDAO == null)
        {
          _dataDAO = newProcessInstanceDataDAO();
        }
return _dataDAO;
      }
    }

    #endregion

    #region Constructors

///<summary>
/// Default Constructor
///</summary>
public ProcessInstanceDAO()
```

```
        {
this._dfParser = newDataFieldsParser();
        }

        #endregion

        #region Public Methods

///<summary>
/// Gets a Process Instance by Id
///</summary>
///<param name="id"></param>
///<returns></returns>
publicProcessInstance GetById(string id)
        {
List<ProcessInstance> lst = this.GetByParams(id, string.Empty, string.Empty, StatusEnum.NONE,
string.Empty);
if (lst[0] != null)
            {
               lst[0].DataFields = this.DataDAO.GetByProcessInstanceId(lst[0].Id);
return lst[0];
            }
returnnull;
        }

///<summary>
/// Gets a process instance by Name
///</summary>
///<param name="name"></param>
///<returns></returns>
publicProcessInstance GetByName(string name)
        {
List<ProcessInstance> lst = this.GetByParams(string.Empty, string.Empty, name, StatusEnum.NONE,
string.Empty);
if (lst[0] != null)
            {
               lst[0].DataFields = this.DataDAO.GetByProcessInstanceId(lst[0].Id);
return lst[0];
            }
returnnull;
        }

///<summary>
/// Gets Process Instances by Process definition id and status
///</summary>
///<param name="procDefId"></param>
///<param name="s"></param>
///<returns></returns>
```

```
publicList<ProcessInstance> GetByProcDefId(string procDefId, StatusEnum s)
      {
returnthis.GetByParams(string.Empty, procDefId, string.Empty, s, string.Empty);
      }

///<summary>
/// Saves the Process instance to the database
///</summary>
///<param name="i"></param>
publicvoid Save(ProcessInstance i)
      {
this._dfParser.DataFields = i.DataFields;
string xml = this._dfParser.GetXML();
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
         {
            conn.Open();
using (SqlCommand cmd = newSqlCommand("ProcInst_Save", conn))
            {
               cmd.CommandType = System.Data.CommandType.StoredProcedure;
               cmd.Parameters.Add(newSqlParameter("@ProcInstId", i.Id));
               cmd.Parameters.Add(newSqlParameter("@ProcDefId", i.ProcDefId));
               cmd.Parameters.Add(newSqlParameter("@Initiator", i.Initiator.Id));
               cmd.Parameters.Add(newSqlParameter("@Status", i.Status.ToString()));
               cmd.Parameters.Add(newSqlParameter("@Name", i.Name));
if (i.CompletedDate.HasValue)
               {
if (i.CompletedDate.Value >DateTime.MinValue)
                  {
                     cmd.Parameters.Add(newSqlParameter("@CompletedDate", i.CompletedDate));
                  }
               }
               cmd.Parameters.Add(newSqlParameter("@XML", xml));
               cmd.ExecuteNonQuery();
            }
         }
      }

      #endregion

      #region Private Methods

///<summary>
/// Creates the process definition from the sql data reader that is
/// returned from the query
///</summary>
///<param name="reader"></param>
///<returns></returns>
privateProcessInstance CreateFromDataReader(SqlDataReader reader)
```

```
        {
ProcessInstance p = newProcessInstance();
            p.Id = reader.IsDBNull(reader.GetOrdinal("ProcInstId")) ? string.Empty :
((Guid)reader["ProcInstId"]).ToString();
            p.ProcDefId = reader.IsDBNull(reader.GetOrdinal("ProcDefId")) ? string.Empty :
((Guid)reader["ProcDefId"]).ToString();
            p.Initiator.Id = reader.IsDBNull(reader.GetOrdinal("Initiator")) ? string.Empty :
((string)reader["Initiator"]).ToString();
            p.Status = reader.IsDBNull(reader.GetOrdinal("Status")) ? StatusEnum.NONE :
EnumTranslator.GetStatus((string)reader["Status"]);
            p.Name = reader.IsDBNull(reader.GetOrdinal("Name")) ? string.Empty : (string)reader["Name"];
            p.CreatedDate = reader.IsDBNull(reader.GetOrdinal("CreatedDate")) ? DateTime.MinValue :
(DateTime)reader["CreatedDate"];
            p.CompletedDate = reader.IsDBNull(reader.GetOrdinal("CompletedDate")) ? DateTime.MinValue
: (DateTime)reader["CompletedDate"];
return p;
        }

///<summary>
/// Queries for activity instances
///</summary>
///<param name="id"></param>
///<param name="procInstId"></param>
///<param name="name"></param>
///<param name="s"></param>
///<returns></returns>
privateList<ProcessInstance> GetByParams(string id, string procDefId, string name, StatusEnum s, string
initiator)
        {
List<ProcessInstance> lst = newList<ProcessInstance>();
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
            {
                conn.Open();
using (SqlCommand cmd = newSqlCommand("ProcInst_GetByParams", conn))
                {
if (!string.IsNullOrEmpty(id))
                    {
                        cmd.Parameters.Add(newSqlParameter("@ProcInstId", id));
                    }
if (!string.IsNullOrEmpty(procDefId))
                    {
                        cmd.Parameters.Add(newSqlParameter("@ProcDefId", procDefId));
                    }
if (!string.IsNullOrEmpty(initiator))
                    {
                        cmd.Parameters.Add(newSqlParameter("@Initiator", initiator));
                    }
if (s != StatusEnum.NONE)
```

```
                {
                    cmd.Parameters.Add(newSqlParameter("@Status", s.ToString()));
                }
if (!string.IsNullOrEmpty(name))
                {
                    cmd.Parameters.Add(newSqlParameter("@Name", name));
                }
                cmd.CommandType = System.Data.CommandType.StoredProcedure;
using (SqlDataReader reader = cmd.ExecuteReader())
                {
if (reader.HasRows)
                    {
while (reader.Read())
                        {
                            lst.Add(CreateFromDataReader(reader));
                        }
                    }
                }
            }
        }

return lst;
      }

      #endregion
   }
}


using System.Collections.Generic;
using System.Data.SqlClient;
using OOWF.Lib.DAL.Base;
using OOWF.Lib.Entities.Definition;
using OOWF.Lib.Shared.Utils;

namespace OOWF.Lib.DAL
{
///<summary>
/// Saves and retrieves process instance data records
///</summary>
publicclassProcessInstanceDataDAO : DAOBase
   {
      #region Instance Data

privateDataFieldsParser _dfParser;

      #endregion
```

```
        #region Constructors

///<summary>
/// Default Constructor
///</summary>
public ProcessInstanceDataDAO()
        {
this._dfParser = newDataFieldsParser();
        }

        #endregion

        #region Public Methods

///<summary>
/// Gets Process Instance Data By Process Instance Id
///</summary>
///<param name="id"></param>
///<returns></returns>
publicList<DataField> GetByProcessInstanceId(string procInstId)
        {
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
            {
                conn.Open();
using (SqlCommand cmd = newSqlCommand("ProcInstData_GetByProcInstId", conn))
                {
                    cmd.Parameters.Add(newSqlParameter("@ProcInstId", procInstId));
                    cmd.CommandType = System.Data.CommandType.StoredProcedure;
using (SqlDataReader reader = cmd.ExecuteReader())
                    {
if (reader.HasRows)
                        {
                            reader.Read();
return CreateFromDataReader(reader);
                        }
                    }
                }
            }
returnnull;
        }

///<summary>
/// Saves the process instance data to the database
///</summary>
///<param name="a"></param>
publicvoid Save(string procInstId, List<DataField> l)
        {
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
```

```
        {
            conn.Open();
using (SqlCommand cmd = newSqlCommand("ProcInstData_Save", conn))
            {
this._dfParser.DataFields = l;
string xml = this._dfParser.GetXML();

                cmd.CommandType = System.Data.CommandType.StoredProcedure;
                cmd.Parameters.Add(newSqlParameter("@ProcInstId", procInstId));
if (!string.IsNullOrEmpty(xml))
                {
                    cmd.Parameters.Add(newSqlParameter("@XML", xml));
                }
                cmd.ExecuteNonQuery();
            }
        }
    }

    #endregion

    #region Private Methods

///<summary>
/// Creates the process instance datafields list from the sql data reader that is
/// returned from the query
///</summary>
///<param name="reader"></param>
///<returns></returns>
privateList<DataField> CreateFromDataReader(SqlDataReader reader)
    {
string xml = reader.IsDBNull(reader.GetOrdinal("XML")) ? string.Empty : (string)reader["XML"];
if (!string.IsNullOrEmpty(xml))
        {
returnthis._dfParser.Parse(xml);
        }
returnnewList<DataField>();
    }

    #endregion
  }
}


using System.Collections.Generic;
using System.Data.SqlClient;
using OOWF.Lib.DAL.Base;
using OOWF.Lib.Entities.Runtime;
```

```
namespace OOWF.Lib.DAL
{
///<summary>
/// Saves and retrieves role records from the database
///</summary>
publicclassRoleDAO : DAOBase
  {
    #region Constructors

///<summary>
/// Default Constructor
///</summary>
public RoleDAO() { }

    #endregion

    #region Public Methods

///<summary>
/// Retrieves a role record by Id
///</summary>
///<param name="userId"></param>
///<returns></returns>
publicRole GetById(string id)
    {
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
      {
        conn.Open();
using (SqlCommand cmd = newSqlCommand("Role_GetById", conn))
        {
          cmd.Parameters.Add(newSqlParameter("@RoleId", id));
          cmd.CommandType = System.Data.CommandType.StoredProcedure;
using (SqlDataReader reader = cmd.ExecuteReader())
          {
if (reader.HasRows)
            {
              reader.Read();
return CreateFromDataReader(reader);
            }
          }
        }
      }
returnnull;
    }

///<summary>
/// Gets all roles for a user
///</summary>
```

```
///<param name="username"></param>
///<returns></returns>
publicList<Role> GetAllForUser(string username)
      {
List<Role> roles = newList<Role>();
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
         {
            conn.Open();
using (SqlCommand cmd = newSqlCommand("Role_GetAllForUser", conn))
            {
               cmd.Parameters.Add(newSqlParameter("@UserId", username));
               cmd.CommandType = System.Data.CommandType.StoredProcedure;
using (SqlDataReader reader = cmd.ExecuteReader())
               {
if (reader.HasRows)
                  {
while (reader.Read())
                     {
                        roles.Add(CreateFromDataReader(reader));
                     }
                  }
               }
            }
         }
return roles;
      }

///<summary>
/// Saves a Role to the database
///</summary>
///<param name="u"></param>
publicvoid Save(Role r)
      {
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
         {
            conn.Open();
using (SqlCommand cmd = newSqlCommand("Role_Save", conn))
            {
               cmd.CommandType = System.Data.CommandType.StoredProcedure;
               cmd.Parameters.Add(newSqlParameter("@RoleId", r.Id));
               cmd.Parameters.Add(newSqlParameter("@DisplayName", r.DisplayName));
               cmd.ExecuteNonQuery();
            }
         }
      }

      #endregion
```

```
        #region Private Methods

///<summary>
/// Creates a role object from a sql data reader
///</summary>
///<param name="reader"></param>
///<returns></returns>
privatestaticRole CreateFromDataReader(SqlDataReader reader)
        {
Role r = newRole();
            r.Id = reader.IsDBNull(reader.GetOrdinal("RoleId")) ? string.Empty : (string)reader["RoleId"];
            r.Name = r.Id;
            r.DisplayName = reader.IsDBNull(reader.GetOrdinal("DisplayName")) ? string.Empty :
(string)reader["DisplayName"];

return r;
        }

        #endregion
    }
}


using System.Collections.Generic;
using System.Data.SqlClient;
using OOWF.Lib.DAL.Base;
using OOWF.Lib.Entities.Runtime;

namespace OOWF.Lib.DAL
{
///<summary>
/// Retrieves and Saves user records
///</summary>
publicclassUserDAO : DAOBase
    {
        #region Constructors

///<summary>
/// Default Constructor
///</summary>
public UserDAO() { }

        #endregion

        #region Public Methods

///<summary>
/// Retrieves a user record by user Id
```

```
///</summary>
///<param name="userId"></param>
///<returns></returns>
publicUser GetById(string userId)
      {
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
         {
            conn.Open();
using (SqlCommand cmd = newSqlCommand("User_GetById", conn))
            {
               cmd.Parameters.Add(newSqlParameter("@UserId", userId));
               cmd.CommandType = System.Data.CommandType.StoredProcedure;
using (SqlDataReader reader = cmd.ExecuteReader())
               {
if (reader.HasRows)
                  {
                     reader.Read();
return CreateFromDataReader(reader);
                  }
               }
            }
         }
returnnull;
      }

///<summary>
/// Gets all users in a role
///</summary>
///<param name="roleId"></param>
///<returns></returns>
publicList<User> GetAllForRole(string roleId)
      {
List<User> users = newList<User>();
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
         {
            conn.Open();
using (SqlCommand cmd = newSqlCommand("User_GetAllForRole", conn))
            {
               cmd.Parameters.Add(newSqlParameter("@RoleId", roleId));
               cmd.CommandType = System.Data.CommandType.StoredProcedure;
using (SqlDataReader reader = cmd.ExecuteReader())
               {
if (reader.HasRows)
                  {
while (reader.Read())
                     {
                        users.Add(CreateFromDataReader(reader));
                     }
```

```
                }
            }
        }
    }
return users;
    }

///<summary>
/// Saves a User to the database
///</summary>
///<param name="u"></param>
publicvoid Save(User u)
    {
using (SqlConnection conn = newSqlConnection(this.ConnectionString))
        {
            conn.Open();
using (SqlCommand cmd = newSqlCommand("User_Save", conn))
            {
                cmd.CommandType = System.Data.CommandType.StoredProcedure;
                cmd.Parameters.Add(newSqlParameter("@UserId", u.Id));
                cmd.Parameters.Add(newSqlParameter("@DisplayName", u.DisplayName));
                cmd.Parameters.Add(newSqlParameter("@Email", u.Email));
                cmd.ExecuteNonQuery();
            }
        }
    }

    #endregion

    #region Private Methods

///<summary>
/// Creates a user object from a sql data reader
///</summary>
///<param name="reader"></param>
///<returns></returns>
privatestaticUser CreateFromDataReader(SqlDataReader reader)
    {
User u = newUser();
        u.Id = reader.IsDBNull(reader.GetOrdinal("UserId")) ? string.Empty : (string)reader["UserId"];
        u.Name = u.Id;
        u.DisplayName = reader.IsDBNull(reader.GetOrdinal("DisplayName")) ? string.Empty :
(string)reader["DisplayName"];
        u.Email = reader.IsDBNull(reader.GetOrdinal("Email")) ? string.Empty : (string)reader["Email"];

return u;
    }
```

```
    #endregion
  }
}


using System;
using System.Collections.Generic;
using OOWF.Lib.Entities.Definition;
using OOWF.Lib.Entities.Runtime;
using OOWF.Lib.Shared.Enums;
using OOWF.Lib.Shared.Utils;

namespace OOWF.Lib.Engine
{
///<summary>
/// The workflow engine
///</summary>
publicclassWFEngine
  {
    #region Instance Data

privateProcessDefinition _currentProcessDef;
privateProcessInstance _currentProcessInst;

    #endregion

    #region Private Properties

///<summary>
/// The current cached process defintion
///</summary>
privateProcessDefinition CurrentDefinition
    {
get
      {
if (this._currentProcessDef == null)
        {
this._currentProcessDef = newProcessDefinition();
        }
returnthis._currentProcessDef;
      }
set
      {
this._currentProcessDef = value;
      }
    }

///<summary>
```

```
/// The current cached process instance
///</summary>
privateProcessInstance CurrentProcessInstance
    {
get
        {
if (this._currentProcessInst == null)
            {
this._currentProcessInst = newProcessInstance();
            }
returnthis._currentProcessInst;
        }
set
        {
this._currentProcessInst = value;
        }
    }

    #endregion

    #region Constructors

///<summary>
/// Default Constructor
///</summary>
public WFEngine() { }

    #endregion

    #region Public Methods

///<summary>
/// Cancel a process instance
///</summary>
///<param name="instId"></param>
///<param name="s"></param>
publicvoid CancelProcessInstance(string instId)
    {
this.ValidateProcessInstanceCache(instId);
this.CurrentProcessInstance.CompletedDate = DateTime.Now;
this.CurrentProcessInstance.Status = StatusEnum.CANCELLED;
this.CurrentProcessInstance.Save();
    }

///<summary>
/// Completes a manual task
///</summary>
///<param name="taskId"></param>
```

```
publicvoid CompleteTask(string taskId)
      {
// Get task from db
ManualTaskInstance m = ManualTaskInstance.GetById(taskId);

this.CompleteTask(m);
      }

///<summary>
/// Completes a manual task
///</summary>
///<param name="m"></param>
publicvoid CompleteTask(ManualTaskInstance m)
      {
// Set the current process instance and definition
this.ValidateProcessInstanceCache(m.ProcessInstanceId);

// update task data
         m.Status = StatusEnum.COMPLETED;
         m.CompletedDate = DateTime.Now;

// Save back to database
         m.Save();

// Traverse the tree
this.Traverse(m);
      }

///<summary>
/// Creates a new process instance
///</summary>
///<param name="definitionName"></param>
///<param name="initiator"></param>
///<returns></returns>
publicProcessInstance CreateProcessInstance(string definitionName, string initiator)
      {
return CreateProcessInstance(definitionName, initiator, null);
      }

///<summary>
/// Creates a new process instance
///</summary>
///<param name="definitionName"></param>
///<param name="initiator"></param>
///<param name="initialValues"></param>
///<returns></returns>
publicProcessInstance CreateProcessInstance(string definitionName, string initiator, Dictionary<string,
string> initialValues)
```

```
    {
// Cache the process definition if needed
this.ValidateDefinitionCacheByName(definitionName);

// Create the new process instance
ProcessInstance inst = newProcessInstance();

        inst.ProcessDefinition = this.CurrentDefinition;
        inst.ProcDefId = this.CurrentDefinition.Id;
        inst.ProcDefName = this.CurrentDefinition.Name;
        inst.Initiator.Id = initiator;
        inst.Name = inst.ProcDefName + "-" + inst.Id;
        inst.DataFields = inst.ProcessDefinition.Process.DataFields;

// Cache this process instance
this.CurrentProcessInstance = inst;

// Update it's datafields
this.UpdateCurrentDataFields(inst.Id, initialValues);

// Execute it
        inst.Execute();

// Get the start activites and traverse them
foreach (Activity a in inst.ProcessDefinition.Process.Activites)
        {
if (a.IncomingTransitions.Count == 0)
          {
this.ProcessActivity(a);
          }
        }

return inst;
    }

///<summary>
/// Retrieves a data field from the current process
///</summary>
///<param name="id"></param>
///<returns></returns>
publicDataField GetDataField(string name)
    {
foreach (DataField d inthis.CurrentProcessInstance.DataFields)
        {
if (d.Name == name)
          {
return d;
          }
```

```
        }
returnnull;
      }

///<summary>
/// Updates a process instances datafields
///</summary>
///<param name="procInstId"></param>
///<param name="updatedValues"></param>
publicvoid UpdateDataFields(string procInstId, Dictionary<string, string> updatedValues)
      {
// Update the fields
this.UpdateCurrentDataFields(procInstId, updatedValues);

// Save the instance
this.CurrentProcessInstance.Save();
      }

      #endregion

      #region Private Methods

///<summary>
/// Parses out the lookup key
///</summary>
///<param name="lookup"></param>
///<returns></returns>
privatestring ParseLookupField(string lookup)
      {
int startIndex = lookup.IndexOf('{');
int endIndex = lookup.LastIndexOf('}') - 1;
if (startIndex > 0 && endIndex > 0 && endIndex > startIndex)
        {
return lookup.Substring(startIndex + 1, (endIndex - startIndex));
        }
return lookup;
      }

///<summary>
/// Parses an expression into a condition evaluator
///</summary>
///<param name="expression"></param>
///<returns></returns>
privateConditionEvaluator ParseConditionExpression(string expression)
      {
ConditionEvaluator c = null;
string[] parts = expression.Split(' ');
if (parts.Length == 3)
```

```
        {
            c = newConditionEvaluator();
            c.LeftData = this.ResolveLookupField(parts[0]);
            c.Operand = c.GetOperator(parts[1]);
            c.RightData = this.ResolveLookupField(parts[2]);
        }
return c;
    }

///<summary>
/// Determines the activity type and processes it
///</summary>
///<param name="a"></param>
privatevoid ProcessActivity(Activity a)
    {
if (a isEventActivity)
        {
this.ProcessEventActivity((EventActivity)a);
        }
elseif (a isManualTask)
        {
this.ProcessManualTask((ManualTask)a);
        }
elseif (a isEmailTask)
        {
this.ProcessEmailActivity((EmailTask)a);
        }
elseif (a isRouteActivity)
        {
this.ProcessRouteActivity((RouteActivity)a);
        }
else
        {
thrownewArgumentException("Unknown Activity Type: " + a.GetType().ToString());
        }
    }

///<summary>
/// Processes an Email Activity from its definition
///</summary>
///<param name="source"></param>
///<param name="i"></param>
privatevoid ProcessEmailActivity(EmailTask source)
    {
EmailTaskInstance i = i = newEmailTaskInstance(this.CurrentProcessInstance.Id, source.Name,
source.DisplayName);
        i.To = "edpeloquin@hotmail.com";
        i.From = "edpeloquin@hotmail.com";
```

```
            i.Subject = source.Subject;
            i.Body = source.Body;
            i.Execute();
this.Traverse(i);
        }

///<summary>
/// Processes an Event Activity
///</summary>
///<param name="source"></param>
///<param name="i"></param>
privatevoid ProcessEventActivity(EventActivity source)
        {
ActivityInstance i = newActivityInstance(this.CurrentProcessInstance.Id, source.Name,
source.DisplayName);
            i.Execute();
this.Traverse(i);
        }

///<summary>
/// Processes a manual task activity from it's definition
///</summary>
///<param name="source"></param>
///<param name="i"></param>
privatevoid ProcessManualTask(ManualTask source)
        {
// Get the participant
Participant p = source.Participants[0];
if (p != null)
            {
DataField d = this.ResolveLookupField(source.Participants[0].Name);
if (d != null)
                {
ManualTaskInstance i = newManualTaskInstance(this.CurrentProcessInstance.Id, source.Name,
source.DisplayName, d.Value.ToString());
                    i.Execute();
                }
            }
        }

///<summary>
/// Processes a route activity from it's definition
///</summary>
///<param name="source"></param>
privatevoid ProcessRouteActivity(RouteActivity source)
        {
// Currently treat all gateways as exclusive
Transition otherTrans = null;
```

```
Activity a = null;
// Get the gateway transitions
foreach (Transition t in source.OutgoingTransistions)
        {
// Save the otherwise branch
if (t.TransitionType == TransitionTypeEnum.OTHERWISE)
           {
              otherTrans = t;
           }
else
           {
// Check the expression and if true then traverse
if (!string.IsNullOrEmpty(t.Expression))
              {
ConditionEvaluator c = this.ParseConditionExpression(t.Expression);
if (c != null)
                {
if (c.Evaluate())
                   {
                      a = (Activity)this.CurrentDefinition.Process.ActivitiesHt[t.To];
break;
                   }
                }
              }
else
              {
                 a = (Activity)this.CurrentDefinition.Process.ActivitiesHt[t.To];
break;
              }
           }

        }
if (a != null)
        {
this.ProcessActivity(a);
        }
elseif (otherTrans != null)
        {
           a = (Activity)this.CurrentDefinition.Process.ActivitiesHt[otherTrans.To];
this.ProcessActivity(a);
        }
     }

///<summary>
/// Resolves a key lookup field ($DATA{}, $USER{}, $ROLE{})
///</summary>
///<param name="lookup"></param>
///<returns></returns>
```

```
privateDataField ResolveLookupField(string lookup)
      {
// Determine the lookup type
string lookupKey = this.ParseLookupField(lookup);
DataField d = null;
if (lookup.Contains("$DATA"))
         {
// Get the key field name
            d = GetDataField(lookupKey);
         }
elseif (lookup.Contains("$USER"))
         {
User u = User.GetById(lookupKey);
if (u != null)
            {
               d = newDataField();
               d.Id = u.Id;
               d.Value = u.Id;
               d.DataType = DataTypeEnum.STRING;
            }
         }
elseif (lookup.Contains("$ROLE"))
         {
List<User> users = User.GetAllForRole(lookupKey);
if (users != null)
            {
if (users.Count > 0)
               {
                  d = newDataField();
                  d.Id = users[0].Id;
                  d.Value = users[0].Id;
                  d.DataType = DataTypeEnum.STRING;
               }
            }
         }
else
         {
            d = newDataField();
            d.Value = lookup;
            d.DataType = ConditionEvaluator.ResolveDataType(lookup);
         }
// Not a lookup field
return d;
      }

///<summary>
/// Updates the current running processes data fields
///</summary>
```

```
///<param name="procInstId"></param>
///<param name="updatedValues"></param>
privatevoid UpdateCurrentDataFields(string procInstId, Dictionary<string, string> updatedValues)
    {
// Validate the cache
this.ValidateProcessInstanceCache(procInstId);

if (updatedValues.Count > 0)
        {
foreach (DataField d inthis.CurrentProcessInstance.DataFields)
          {
if (updatedValues.ContainsKey(d.Name))
            {
               d.Value = updatedValues[d.Name];
            }
          }
        }
    }

///<summary>
/// Determines the outgoing transitions for this activity and processes them
///</summary>
///<param name="completedActivity"></param>
privatevoid Traverse(ActivityInstance completedActivity)
    {
// Get the id of the current activity in the definition
string lookupId = this.CurrentDefinition.Process.ActivityNames[completedActivity.Name];

// Get the activity object
Activity a = (Activity)this.CurrentDefinition.Process.ActivitiesHt[lookupId];

// Check the outgoing transitions
if (a.OutgoingActivityIds.Count > 0)
        {
List<Activity> outGoingActivities = newList<Activity>();
// Get the activty objects from the definition
foreach (string id in a.OutgoingActivityIds)
          {
             outGoingActivities.Add((Activity)this.CurrentDefinition.Process.ActivitiesHt[id]);
          }

// Process the outgoing activities
foreach (Activity a1 in outGoingActivities)
          {
this.ProcessActivity(a1);
          }
        }
elseif (a isEventActivity)
```

```
        {
if (((EventActivity)a).EventType == EventTypeEnum.END)
          {
// Stop the process instance because the end event was hit
this.CurrentProcessInstance.CompletedDate = DateTime.Now;
this.CurrentProcessInstance.Status = StatusEnum.COMPLETED;
this.CurrentProcessInstance.Save();
          }
        }
      }

///<summary>
/// Retrieves the latest version of a process definition if needed
///</summary>
///<param name="defName"></param>
privatevoid ValidateDefinitionCacheByName(string defName)
      {
// Cache the definition if needed
if (this.CurrentDefinition.Name != defName)
        {
this.CurrentDefinition = ProcessDefinition.GetByName(defName);
        }
      }

///<summary>
/// Retrieves the process definition if needed
///</summary>
///<param name="defId"></param>
privatevoid ValidateDefinitionCacheById(string defId)
      {
if (this.CurrentDefinition.Id != defId)
        {
this.CurrentDefinition = ProcessDefinition.GetById(defId);
        }
      }

///<summary>
/// Sets the current process instance
///</summary>
///<param name="instId"></param>
privatevoid ValidateProcessInstanceCache(string instId)
      {
if (this.CurrentProcessInstance.Id != instId)
        {
this.CurrentProcessInstance = ProcessInstance.GetById(instId);
this.ValidateDefinitionCacheById(this.CurrentProcessInstance.ProcDefId);
        }
      }
```

```
        #endregion
    }
}



namespace OOWF.Lib.Entities.Base
{
///<summary>
/// Represents a base entity with all of the common attributes
///</summary>
publicabstractclassBaseEntity
    {
        #region Public Properties

///<summary>
/// The entity Id
///</summary>
publicstring Id { get; set; }

///<summary>
/// The entity name
///</summary>
publicstring Name { get; set; }

///<summary>
/// The entity display name
///</summary>
publicstring DisplayName { get; set; }

        #endregion
    }
}


using System;
using OOWF.Lib.Shared.Enums;

namespace OOWF.Lib.Entities.Base
{
///<summary>
/// Represents a runtime workflow object
///</summary>
publicabstractclassRuntimeEntity : BaseEntity
    {
        #region Public Properties
```

```
///<summary>
/// The process instance this activity belongs to
///</summary>
publicstring ProcessInstanceId { get; set; }

///<summary>
/// The process status
///</summary>
publicStatusEnum Status { get; set; }

///<summary>
/// The process Created date
///</summary>
publicDateTime CreatedDate { get; set; }

///<summary>
/// The process completed date
///</summary>
publicDateTime? CompletedDate { get; set; }

      #endregion

      #region Public Methods

///<summary>
/// Executes this activity
///</summary>
publicabstractvoid Execute();

///<summary>
/// Saves the instance to the database
///</summary>
publicabstractvoid Save();

      #endregion
   }
}


using System;
using System.Collections.Generic;
using OOWF.Lib.Entities.Base;
using OOWF.Lib.Shared.Enums;

namespace OOWF.Lib.Entities.Definition
{
///<summary>
/// Represents an activity in the workflow
```

```
///</summary>
publicclassActivity : BaseEntity
  {
    #region Public Properties

///<summary>
/// A description of the activity
///</summary>
publicstring Description { get; set; }

///<summary>
/// The Activity Start Date
///</summary>
publicDateTime StartDate { get; set; }

///<summary>
/// The Activity Complete Date
///</summary>
publicDateTime? CompletedDate { get; set; }

///<summary>
/// The current status of the activity
///</summary>
publicStatusEnum Status { get; set; }

///<summary>
/// The List of Participants for this activity
///</summary>
publicList<Participant> Participants { get; set; }

///<summary>
/// A list of all incoming activity ids
///</summary>
publicList<string> IncomingActivityIds { get; set; }

///<summary>
/// A list of all outgoing activity ids
///</summary>
publicList<string> OutgoingActivityIds { get; set; }

///<summary>
/// A list of all incoming transitions
///</summary>
publicList<Transition> IncomingTransitions { get; set; }

///<summary>
/// A list of all outgoing transitions
///</summary>
```

```
publicList<Transition> OutgoingTransistions { get; set; }

      #endregion

      #region Constructors

///<summary>
/// Default Constructor
///</summary>
public Activity()
      {
this.Status = StatusEnum.NONE;
this.Participants = newList<Participant>();
this.IncomingActivityIds = newList<string>();
this.OutgoingActivityIds = newList<string>();
this.IncomingTransitions = newList<Transition>();
this.OutgoingTransistions = newList<Transition>();
      }

///<summary>
/// Overloaded Constructor
///</summary>
///<param name="id"></param>
///<param name="name"></param>
///<param name="displayName"></param>
public Activity(string id, string name, string displayName)
      {
this.Id = id;
this.Name = name;
this.DisplayName = displayName;
this.Status = StatusEnum.NONE;
this.Participants = newList<Participant>();
      }

      #endregion
   }
}
using OOWF.Lib.Entities.Base;
using OOWF.Lib.Shared.Enums;

namespace OOWF.Lib.Entities.Definition
{
publicclassDataField : BaseEntity
   {
      #region Instance Data

privateobject _value;
```

```
        #endregion

        #region Public Properties

///<summary>
/// The Datafield's type
///</summary>
publicDataTypeEnum DataType { get; set; }


///<summary>
/// The datafield's value
///</summary>
publicobject Value
        {
get
            {
returnthis._value;
            }
set
            {
this._value = value;
            }
        }

        #endregion

        #region Constructors

///<summary>
/// Default Constructor
///</summary>
public DataField() { }

///<summary>
/// Overloaded Constructor
///</summary>
///<param name="id"></param>
///<param name="name"></param>
///<param name="displayName"></param>
///<param name="type"></param>
public DataField(string id, string name, string displayName, DataTypeEnum type, object value)
        {
this.Id = id;
this.Name = name;
this.DisplayName = displayName;
this.DataType = type;
this.Value = value;
```

```
        }

        #endregion

        #region Public Methods

///<summary>
/// Translates a type text into an enumeration value
///</summary>
///<param name="typeText"></param>
///<returns></returns>
publicDataTypeEnum GetDataType(string typeText)
        {
switch (typeText.Trim().ToLower())
            {
case"boolean":
returnDataTypeEnum.BOOLEAN;
case"float":
returnDataTypeEnum.FLOAT;
case"integer":
returnDataTypeEnum.INTEGER;
default:
returnDataTypeEnum.STRING;
            }
        }

        #endregion

        #region Private Methods

privatevoid SetValue(object value)
        {
switch (this.DataType)
            {
caseDataTypeEnum.BOOLEAN:
this.Value = (bool)value;
break;
caseDataTypeEnum.FLOAT:
this.Value = (float)value;
break;
caseDataTypeEnum.INTEGER:
this.Value = (int)value;
break;
caseDataTypeEnum.STRING:
this.Value = (string)value;
break;
default:
this.Value = null;
```

```
break;
        }
      }

      #endregion
   }
}


namespace OOWF.Lib.Entities.Definition
{
///<summary>
/// Represents an Email Task
///</summary>
publicclassEmailTask : Activity
   {
      #region Public Properties

///<summary>
/// Message From
///</summary>
publicstring From { get; set; }

///<summary>
/// Message To
///</summary>
publicstring To { get; set; }

///<summary>
/// Message Body
///</summary>
publicstring Body { get; set; }

///<summary>
/// Message Subject
///</summary>
publicstring Subject { get; set; }

      #endregion

      #region Constructors

///<summary>
/// Default Constructor
///</summary>
public EmailTask() { }

///<summary>
```

```
/// Overloaded Constructor
///</summary>
///<param name="from"></param>
///<param name="to"></param>
///<param name="body"></param>
///<param name="subject"></param>
public EmailTask(string from, string to, string body, string subject)
      {
this.From = from;
this.To = to;
this.Body = body;
this.Subject = subject;
      }

      #endregion
   }
}

using OOWF.Lib.Shared.Enums;

namespace OOWF.Lib.Entities.Definition
{
///<summary>
/// Represents an event activity
///</summary>
publicclassEventActivity : Activity
   {
      #region Public Properties

///<summary>
/// The type of Event Activity
///</summary>
publicEventTypeEnum EventType { get; set; }

      #endregion

      #region Constructors

///<summary>
/// Default Constructor
///</summary>
public EventActivity() { }

///<summary>
/// Overloaded Constructor
///</summary>
///<param name="type"></param>
public EventActivity(EventTypeEnum type)
```

```
        {
this.EventType = type;
        }

        #endregion
    }
}


namespace OOWF.Lib.Entities.Definition
{
publicclassManualTask : Activity
    {
        #region Constructors

///<summary>
/// Default Constructor
///</summary>
public ManualTask() { }

        #endregion
    }
}

using OOWF.Lib.Entities.Base;
using OOWF.Lib.Shared.Enums;

namespace OOWF.Lib.Entities.Definition
{
publicclassParticipant : BaseEntity
    {
        #region Public Properties

///<summary>
/// The participant type
///</summary>
publicParticipantTypeEnum Type { get; set; }

        #endregion

        #region Constructors

///<summary>
/// Default Constructor
///</summary>
public Participant() { }

///<summary>
```

```
/// Overloaded Constructor
///</summary>
///<param name="id"></param>
///<param name="name"></param>
///<param name="displayName"></param>
///<param name="type"></param>
public Participant(string id, string name, string displayName, ParticipantTypeEnum type)
      {
this.Id = id;
this.Name = name;
this.DisplayName = displayName;
this.Type = type;
      }

      #endregion
   }
}

using System.Collections.Generic;
using System.Collections;
using OOWF.Lib.Entities.Base;
using OOWF.Lib.Shared.Enums;

namespace OOWF.Lib.Entities.Definition
{
///<summary>
/// Represents a process definition
///</summary>
publicclassProcess : BaseEntity
   {
      #region Public Properties

///<summary>
/// The process author
///</summary>
publicstring Author { get; set; }

///<summary>
/// The process creation date
///</summary>
publicstring CreatedDate { get; set; }

///<summary>
/// The process description
///</summary>
publicstring Description { get; set; }

///<summary>
```

```
/// The process status
///</summary>
publicStatusEnum  Status { get; set; }

///<summary>
/// The process version
///</summary>
publicstring version { get; set; }

///<summary>
/// The process xml definition
///</summary>
publicstring XMLDefinition { get; set; }

///<summary>
/// The process datafields
///</summary>
publicList<DataField> DataFields { get; set; }

///<summary>
/// The process participants
///</summary>
publicList<Participant> Participants { get; set; }

///<summary>
/// The process activities
///</summary>
publicList<Activity> Activites { get; set; }

publicHashtable ActivitiesHt { get; set; }

///<summary>
/// Name value lookup from activity name to id
///</summary>
publicDictionary<string, string> ActivityNames { get; set; }

///<summary>
/// The process transitions
///</summary>
publicList<Transition> Transitions { get; set; }

    #endregion

    #region Constructors

///<summary>
/// Default Constructor
///</summary>
```

```
public Process()
      {
this.Status = StatusEnum.ACTIVE;
this.DataFields = newList<DataField>();
this.Participants = newList<Participant>();
this.Activites = newList<Activity>();
this.Transitions = newList<Transition>();
this.ActivitiesHt = newHashtable();
this.ActivityNames = newDictionary<string, string>();
      }

///<summary>
/// Overloaded Constructor
///</summary>
///<param name="id"></param>
///<param name="name"></param>
public Process(string id, string name) : this()
      {
this.Id = id;
this.Name = name;
      }

      #endregion

      #region Public Methods

///<summary>
/// Gets a Participant by it's id
///</summary>
///<param name="id"></param>
///<returns></returns>
publicParticipant GetParticipantById(string id)
      {
if (this.Participants != null)
        {
foreach (Participant p inthis.Participants)
          {
if (p.Id == id)
            {
return p;
            }
          }
        }
returnnull;
      }

      #endregion
  }
```

```
}

using OOWF.Lib.Shared.Enums;

namespace OOWF.Lib.Entities.Definition
{
///<summary>
/// Represents a route activity
///</summary>
publicclassRouteActivity : Activity
  {
    #region Public Properties

///<summary>
/// Gateway type
///</summary>
publicBehaviorTypeEnum GatewayType { get; set; }

    #endregion

    #region Constructors

///<summary>
/// Default Constructor
///</summary>
public RouteActivity()
    {
this.GatewayType = BehaviorTypeEnum.EXCLUSIVE;
    }

    #endregion
  }
}

using OOWF.Lib.Entities.Base;
using OOWF.Lib.Shared.Enums;

namespace OOWF.Lib.Entities.Definition
{
///<summary>
/// Represents a transition
///</summary>
publicclassTransition : BaseEntity
  {
    #region Public Properties

///<summary>
/// From activity Id
```

```
///</summary>
publicstring From { get; set; }

///<summary>
/// To activity Id
///</summary>
publicstring To { get; set; }

///<summary>
/// Transition Type
///</summary>
publicTransitionTypeEnum TransitionType { get; set; }

///<summary>
/// The expression
///</summary>
publicstring Expression { get; set; }

      #endregion

      #region Constructors

///<summary>
/// Default Constructor
///</summary>
public Transition() { }

///<summary>
/// Overloaded Constructor
///</summary>
///<param name="id"></param>
///<param name="name"></param>
///<param name="displayName"></param>
///<param name="fromActivityId"></param>
///<param name="toActivityId"></param>
///<param name="type"></param>
///<param name="expression"></param>
public Transition(string id, string name, string displayName, string fromActivityId, string toActivityId,
TransitionTypeEnum type, string expression)
      {
this.Id = id;
this.Name = name;
this.DisplayName = displayName;
this.From = fromActivityId;
this.To = toActivityId;
      }

      #endregion
```

```
   }
}

using System;
using OOWF.Lib.Entities.Base;
using OOWF.Lib.Entities.Definition;
using OOWF.Lib.DAL;
using OOWF.Lib.Shared.Utils;

namespace OOWF.Lib.Entities.Runtime
{
///<summary>
/// Represents an Activity Instance
///</summary>
publicclassActivityInstance : RuntimeEntity
   {
      #region Instance Data

privatestaticActivityInstanceDAO _dao;

      #endregion

      #region Private Properties

///<summary>
/// This objects data access object
///</summary>
privatestaticActivityInstanceDAO DAO
      {
get
         {
if (_dao == null)
            {
               _dao = newActivityInstanceDAO();
            }
return _dao;
         }
      }

      #endregion

      #region Constructors

///<summary>
/// Default Constructor
///</summary>
public ActivityInstance()
      {
```

```
this.Id = Guid.NewGuid().ToString();
this.Status = Shared.Enums.StatusEnum.ACTIVE;
      }

///<summary>
/// Overloaded Constructor
///</summary>
///<param name="ProcInstId"></param>
public ActivityInstance(string ProcInstId)
         : this()
      {
this.ProcessInstanceId = ProcInstId;
      }

///<summary>
/// Overloaded Constructor 2
///</summary>
///<param name="procInstId"></param>
///<param name="name"></param>
///<param name="displayName"></param>
public ActivityInstance(string procInstId, string name, string displayName) : this(procInstId)
      {
this.Name = name;
this.DisplayName = displayName;
      }

      #endregion

      #region Base Overrides

///<summary>
/// Executes this activity and saves it to the database
///</summary>
publicoverridevoid Execute()
      {
try
         {
Logger.Log(this.DisplayName + " has started executing.", null);
// sleep for a second
this.Status = Shared.Enums.StatusEnum.COMPLETED;
this.CompletedDate = DateTime.Now;
// Saves the record in the database
this.Save();
Logger.Log(this.DisplayName + " has finished executing.", null);
         }
catch (System.Exception e)
         {
Logger.Log(this.DisplayName + " has failed to execute.", e);
```

```
        }
      }

///<summary>
/// Saves this activity instance to the database
///</summary>
publicoverridevoid Save()
      {
        DAO.Save(this);
      }

      #endregion
  }
}


using System;
using System.Net.Mail;
using OOWF.Lib.Shared.Utils;

namespace OOWF.Lib.Entities.Runtime
{
///<summary>
/// Represents an email task instance
///</summary>
publicclassEmailTaskInstance : ActivityInstance
  {
    #region Public Properties

///<summary>
/// Message From
///</summary>
publicstring From { get; set; }

///<summary>
/// Message To
///</summary>
publicstring To { get; set; }

///<summary>
/// Message Body
///</summary>
publicstring Body { get; set; }

///<summary>
/// Message Subject
///</summary>
publicstring Subject { get; set; }
```

```
        #endregion

        #region Constructors

///<summary>
/// Default Constructor
///</summary>
public EmailTaskInstance()
            : base() { }

///<summary>
/// Overloaded constructor
///</summary>
///<param name="from"></param>
///<param name="to"></param>
///<param name="body"></param>
///<param name="subject"></param>
public EmailTaskInstance(string from, string to, string body, string subject)
            : this()
        {
this.From = from;
this.To = to;
this.Body = body;
this.Subject = subject;
        }

///<summary>
/// Overloaded constructor 2
///</summary>
///<param name="procInstId"></param>
///<param name="name"></param>
///<param name="displayName"></param>
public EmailTaskInstance(string procInstId, string name, string displayName) : base(procInstId, name,
displayName) { }

        #endregion

        #region Base Overrides

///<summary>
/// Executes this task
///</summary>
publicoverridevoid Execute()
        {
// Send Email
try
            {
```

```
Logger.Log(this.DisplayName + " has started executing.", null);
if (!string.IsNullOrEmpty(this.From) && !string.IsNullOrEmpty(this.To) &&
                !string.IsNullOrEmpty(this.Body) && !string.IsNullOrEmpty(this.Subject))
            {
MailMessage m = newMailMessage(this.From, this.To, this.Subject, this.Body);
                m.IsBodyHtml = true;
                System.Net.Mail.SmtpClient client = newSmtpClient("localhost");
                client.Send(m);
            }
Logger.Log(this.DisplayName + " has finished executing.", null);
        }
catch (System.Exception e)
        {

Logger.Log(this.DisplayName + " has failed to execute.", e);
        }

this.CompletedDate = DateTime.Now;
this.Status = Shared.Enums.StatusEnum.COMPLETED;

this.Save();
      }

///<summary>
/// Saves this activity to the database
///</summary>
publicoverridevoid Save()
      {
base.Save();
      }

      #endregion
   }
}

using OOWF.Lib.DAL;
using OOWF.Lib.Entities.Definition;
using System.Collections.Generic;
using OOWF.Lib.Shared.Enums;
using OOWF.Lib.Shared.Utils;

namespace OOWF.Lib.Entities.Runtime
{
///<summary>
/// Represents a Manual Task Instance
///</summary>
publicclassManualTaskInstance : ActivityInstance
   {
```

```
        #region Instance Data

privatestaticManualTaskInstanceDAO _dao;

        #endregion

        #region Private Properties

///<summary>
/// This objects data access object
///</summary>
privatestaticManualTaskInstanceDAO DAO
        {
get
            {
if (_dao == null)
                {
                    _dao = newManualTaskInstanceDAO();
                }
return _dao;
            }
        }

        #endregion

        #region Public Properties

///<summary>
/// The performer of the activity
///</summary>
publicUser Performer { get; set; }

        #endregion

        #region Constructors

///<summary>
/// Default Constructor
///</summary>
public ManualTaskInstance()
            : base()
        {
this.Performer = newUser();
        }

///<summary>
/// Overloaded Constructor
///</summary>
```

```
///<param name="procInstId"></param>
///<param name="performer"></param>
public ManualTaskInstance(string procInstId, string performer)
      {
this.Performer = newUser();
this.Performer.Id = performer;
      }

///<summary>
/// Overloaded constructor 2
///</summary>
///<param name="procInstId"></param>
///<param name="name"></param>
///<param name="displayName"></param>
public ManualTaskInstance(string procInstId, string name, string displayName, string performer)
          : base(procInstId, name, displayName)
      {
this.Performer = newUser();
this.Performer.Id = performer;
      }

      #endregion

      #region Public Static Methods

///<summary>
/// Gets a task by it's id
///</summary>
///<param name="id"></param>
///<returns></returns>
publicstaticManualTaskInstance GetById(string id)
      {
return DAO.GetById(id);
      }

///<summary>
/// Gets Manual Tasks by Name
///</summary>
///<param name="procInstId"></param>
///<param name="name"></param>
///<returns></returns>
publicstaticManualTaskInstance GetByName(string procInstId, string name)
      {
return DAO.GetByName(procInstId, name);
      }

///<summary>
/// Gets Manual Tasks by Process instance id
```

```
///</summary>
///<param name="procInstId"></param>
///<returns></returns>
publicstaticList<ManualTaskInstance> GetByProcInstId(string procInstId)
    {
return GetByProcInstId(procInstId, StatusEnum.NONE);
    }

///<summary>
/// Gets Manual Tasks by Process instance id and status
///</summary>
///<param name="procDefId"></param>
///<param name="s"></param>
///<returns></returns>
publicstaticList<ManualTaskInstance> GetByProcInstId(string procInstId, StatusEnum s)
    {
return DAO.GetByProcInstId(procInstId, s);
    }

///<summary>
/// Gets Manual Tasks by Process instance id, status, and performer
///</summary>
///<param name="procInstId"></param>
///<param name="s"></param>
///<param name="performer"></param>
///<returns></returns>
publicstaticList<ManualTaskInstance> GetByPerformer(string procInstId, StatusEnum s, string
performer)
    {
return DAO.GetByPerformer(procInstId, s, performer);
    }

    #endregion

    #region Base Overrides

///<summary>
/// Executes this task
///</summary>
publicoverridevoid Execute()
    {
try
      {
Logger.Log(this.DisplayName + " has started executing.", null);
// Execute this specific activity
this.Save();
Logger.Log(this.DisplayName + " has finished executing.", null);
      }
```

```
catch (System.Exception e)
        {
Logger.Log(this.DisplayName + " has failed to execute.", e);
        }
      }

///<summary>
/// Saves this task to the database
///</summary>
publicoverridevoid Save()
      {
         DAO.Save(this);
      }

      #endregion
   }
}


using System;
using OOWF.Lib.DAL;
using OOWF.Lib.Entities.Base;
using OOWF.Lib.Entities.Definition;
using OOWF.Lib.Shared.Enums;
using OOWF.Lib.Shared.Utils;
using OOWF.Lib.Engine;

namespace OOWF.Lib.Entities.Runtime
{
///<summary>
/// Represents a specific version of a process defintion
///</summary>
publicclassProcessDefinition : RuntimeEntity
   {
      #region Instance Data

privatestaticProcessDefinitionDAO _dao;

      #endregion

      #region Private Properties

privatestaticProcessDefinitionDAO DAO
      {
get
         {
if (_dao == null)
            {
```

```
                _dao = newProcessDefinitionDAO();
            }
return _dao;
        }
    }

    #endregion

    #region Public Properties

///<summary>
/// Author
///</summary>
publicstring Author { get; set; }

///<summary>
/// Description
///</summary>
publicstring Description { get; set; }

///<summary>
/// Version
///</summary>
publicstring Version { get; set; }

///<summary>
/// Process Object Model
///</summary>
publicProcess Process { get; set; }

///<summary>
/// Process XML Definition
///</summary>
publicstring XMLDefinition { get; set; }

    #endregion

    #region Constructors

///<summary>
/// Default Constructor
///</summary>
public ProcessDefinition()
    {
this.Id = Guid.NewGuid().ToString();
    }

///<summary>
```

```
/// Overloaded Constructor
///</summary>
///<param name="xmlDefinition"></param>
public ProcessDefinition(string xmlDefinition)
        : this()
    {
ProcDefXMLParser parser = newProcDefXMLParser(xmlDefinition);
this.Process = parser.Parse();
this.Author = this.Process.Author;
this.Description = this.Process.Description;
this.Name = this.Process.Name;
this.DisplayName = this.Process.DisplayName;
this.Status = StatusEnum.ACTIVE;
this.Version = this.Process.version;
this.XMLDefinition = this.Process.XMLDefinition;
    }

    #endregion

    #region Public Static Methods

///<summary>
/// Gets a process definition by id
///</summary>
///<param name="id"></param>
///<returns></returns>
publicstaticProcessDefinition GetById(string id)
    {
return DAO.GetById(id);
    }

    #endregion

    #region Public Methods

///<summary>
/// Gets a process definition record by it's name
///</summary>
///<param name="definitionName"></param>
///<returns></returns>
publicstaticProcessDefinition GetByName(string definitionName)
    {
return DAO.GetByName(definitionName);
    }

    #endregion

    #region Base Overrides
```

```
///<summary>
/// Executes this runtime entity (not supported here)
///</summary>
publicoverridevoid Execute()
      {
// Not Implemented
      }

///<summary>
/// Saves a process definition record to the database
///</summary>
publicoverridevoid Save()
      {
         DAO.Save(this);
      }

      #endregion
   }
}

using System;
using System.Collections.Generic;
using OOWF.Lib.DAL;
using OOWF.Lib.Entities.Base;
using OOWF.Lib.Entities.Definition;
using OOWF.Lib.Shared.Utils;

namespace OOWF.Lib.Entities.Runtime
{
///<summary>
/// Represents a process instance
///</summary>
publicclassProcessInstance : RuntimeEntity
   {
      #region Instance Data

privatestaticProcessInstanceDAO _dao;

privatestaticProcessInstanceDataDAO _dataDao;

      #endregion

      #region Private Properties

///<summary>
/// This objects data access object
///</summary>
```

```
privatestaticProcessInstanceDAO DAO
    {
get
    {
if (_dao == null)
        {
            _dao = newProcessInstanceDAO();
        }
return _dao;
    }
    }

///<summary>
/// This objects custom attributes data access object
///</summary>
privatestaticProcessInstanceDataDAO DataDAO
    {
get
    {
if (_dataDao == null)
        {
            _dataDao = newProcessInstanceDataDAO();
        }
return _dataDao;
    }
    }

    #endregion

    #region Public Properties

///<summary>
/// The Process Definition Object Model (Used for logic)
///</summary>
publicProcessDefinition ProcessDefinition { get; set; }

///<summary>
/// The name of the process definition
///</summary>
publicstring ProcDefName { get; set; }

///<summary>
/// The process definition id
///</summary>
publicstring ProcDefId { get; set; }

///<summary>
/// The process initiator
```

```
///</summary>
publicUser Initiator { get; set; }

///<summary>
/// The activities of the process
///</summary>
publicList<ActivityInstance> Activities { get; set; }

///<summary>
/// The Runtime datafields
///</summary>
publicList<DataField> DataFields { get; set; }

      #endregion

      #region Constructors

///<summary>
/// Default Constructor
///</summary>
public ProcessInstance()
      {
// Generate a unique Id for this instance
this.Id = Guid.NewGuid().ToString();
this.Status = Shared.Enums.StatusEnum.ACTIVE;
this.Initiator = newUser();
this.Activities = newList<ActivityInstance>();
this.DataFields = newList<DataField>();
      }

///<summary>
/// Overloaded Constructor
///</summary>
///<param name="ProcDefName"></param>
///<param name="initiatior"></param>
public ProcessInstance(string ProcDefName, string initiatior)
         : this()
      {
// Initialize instance data
this.Initiator = newUser() { Id = initiatior };
this.ProcDefName = ProcDefName;
      }

      #endregion

      #region Base Overrides

///<summary>
```

```
/// Executes the process instance
///</summary>
publicoverridevoid Execute()
      {
try
         {
Logger.Log(this.DisplayName + " has started executing.", null);
// Saves the record in the database
this.Save();
Logger.Log(this.DisplayName + " has finished executing.", null);
         }
catch (System.Exception e)
         {
Logger.Log(this.DisplayName + " has failed to execute.", e);
         }
      }

///<summary>
/// Saves a process instance to the database
///</summary>
publicoverridevoid Save()
      {
         DAO.Save(this);
      }

      #endregion

      #region Public Static Methods

///<summary>
/// Gets a process instance by id
///</summary>
///<param name="id"></param>
///<returns></returns>
publicstaticProcessInstance GetById(string id)
      {
return DAO.GetById(id);
      }

      #endregion

      #region Public Methods


      #endregion
   }
}
```

```
using System.Collections.Generic;
using OOWF.Lib.DAL;
using OOWF.Lib.Entities.Base;

namespace OOWF.Lib.Entities.Runtime
{
///<summary>
/// Represents a System Role
///</summary>
publicclassRole : RuntimeEntity
  {
    #region Static Data

privatestaticRoleDAO _dao;

    #endregion

    #region Private Static Properties

privatestaticRoleDAO DAO
    {
get
      {
if (_dao == null)
        {
          _dao = newRoleDAO();
        }
return _dao;
      }
    }

    #endregion

    #region Public Properties

///<summary>
/// The Members of the role
///</summary>
publicList<User> Users { get; set; }

    #endregion

    #region Constructors

///<summary>
/// Default Constructor
///</summary>
public Role()
```

```
        {
this.Users = newList<User>();
        }

///<summary>
/// Overloaded Constructor
///</summary>
///<param name="name"></param>
///<param name="displayName"></param>
public Role(string name, string displayName)
            : this()
        {
this.Id = name;
this.Name = name;
this.DisplayName = displayName;
        }

        #endregion

        #region Public Static Methods

///<summary>
/// Gets a role by it's id
///</summary>
///<param name="id"></param>
///<returns></returns>
publicstaticRole GetById(string id)
        {
return DAO.GetById(id);
        }

///<summary>
/// Gets all roles for a user
///</summary>
///<param name="userId"></param>
///<returns></returns>
publicstaticList<Role> GetAllForUser(string userId)
        {
return DAO.GetAllForUser(userId);
        }

        #endregion

        #region Base Overrides

publicoverridevoid Execute()
        {
thrownew System.NotImplementedException();
```

```
        }

publicoverridevoid Save()
        {
            DAO.Save(this);
        }

        #endregion
    }
}

using System.Collections.Generic;
using OOWF.Lib.DAL;
using OOWF.Lib.Entities.Base;

namespace OOWF.Lib.Entities.Runtime
{
publicclassUser : RuntimeEntity
    {
        #region Static Data

privatestaticUserDAO _dao;

        #endregion

        #region Private Static Properties

privatestaticUserDAO DAO
        {
get
            {
if (_dao == null)
                {
                    _dao = newUserDAO();
                }
return _dao;
            }
        }

        #endregion

        #region Public Properties

///<summary>
/// The user's email address
///</summary>
publicstring Email { get; set; }
```

```
///<summary>
/// The Roles for this user
///</summary>
publicList<Role> Roles { get; set; }

      #endregion

      #region Constructors

///<summary>
/// Default Constructor
///</summary>
public User() { }

///<summary>
/// Overloaded Constructor
///</summary>
///<param name="id"></param>
///<param name="name"></param>
///<param name="displayName"></param>
///<param name="email"></param>
public User(string id, string name, string displayName, string email)
      {
this.Id = id;
this.Name = name;
this.DisplayName = displayName;
this.Email = email;
      }

      #endregion

      #region Public Static Methods

///<summary>
/// Gets a user by id
///</summary>
///<param name="userId"></param>
///<returns></returns>
publicstaticUser GetById(string userId)
      {
return DAO.GetById(userId);
      }

///<summary>
/// Gets all users in a role
///</summary>
///<param name="roleId"></param>
///<returns></returns>
```

```
publicstaticList<User> GetAllForRole(string roleId)
      {
return DAO.GetAllForRole(roleId);
      }

      #endregion

      #region Base Overrides

publicoverridevoid Execute()
      {
thrownew System.NotImplementedException();
      }

///<summary>
/// Saves a user record to the database
///</summary>
publicoverridevoid Save()
      {
         DAO.Save(this);
      }

      #endregion
   }
}


using System;
using System.Text.RegularExpressions;
using OOWF.Lib.Entities.Definition;
using OOWF.Lib.Shared.Enums;

namespace OOWF.Lib.Shared.Utils
{
///<summary>
/// Evaluates simple expressions
///</summary>
publicclassConditionEvaluator
   {
      #region Public Properties

///<summary>
/// Left Data Field
///</summary>
publicDataField LeftData { get; set; }

///<summary>
/// Right Data Field
```

```
///</summary>
publicDataField RightData { get; set; }

///<summary>
/// The expression operand
///</summary>
publicOperandEnum Operand { get; set; }

    #endregion

    #region Constructors

///<summary>
/// Default Constructor
///</summary>
public ConditionEvaluator() { }

///<summary>
/// Overloaded Constructor
///</summary>
///<param name="leftData"></param>
///<param name="leftDataType"></param>
///<param name="rightData"></param>
///<param name="dataType"></param>
///<param name="operand"></param>
public ConditionEvaluator(DataField leftData, DataField rightData, OperandEnum operand)
    {
this.LeftData = leftData;
this.RightData = rightData;
this.Operand = operand;
    }

    #endregion

    #region Public Static Methods

///<summary>
/// Evaluates a boolean expression
///</summary>
///<param name="left"></param>
///<param name="right"></param>
///<returns></returns>
publicstaticbool EvaluateBooleanExpression(bool left, bool right, OperandEnum o)
    {
switch (o)
      {
caseOperandEnum.EQ:
return left == right;
```

```
caseOperandEnum.NEQ:
return left != right;
default:
returnfalse;
        }
      }

///<summary>
/// Evaluate an integer expression
///</summary>
///<param name="left"></param>
///<param name="right"></param>
///<returns></returns>
publicstaticbool EvaluateIntegerExpression(int left, int right, OperandEnum o)
      {
switch (o)
        {
caseOperandEnum.EQ:
return left == right;
caseOperandEnum.GT:
return left > right;
caseOperandEnum.GTEQ:
return left >= right;
caseOperandEnum.LT:
return left < right;
caseOperandEnum.LTEQ:
return left <= right;
caseOperandEnum.NEQ:
return left != right;
default:
returnfalse;
        }
      }

///<summary>
/// Evaluates a float expression
///</summary>
///<param name="left"></param>
///<param name="right"></param>
///<returns></returns>
publicstaticbool EvaluateFloatExpression(decimal left, decimal right, OperandEnum o)
      {
switch (o)
        {
caseOperandEnum.EQ:
return left == right;
caseOperandEnum.GT:
return left > right;
```

```
caseOperandEnum.GTEQ:
return left >= right;
caseOperandEnum.LT:
return left < right;
caseOperandEnum.LTEQ:
return left <= right;
caseOperandEnum.NEQ:
return left != right;
default:
returnfalse;
        }
    }

///<summary>
/// Evaluates a string expression
///</summary>
///<param name="left"></param>
///<param name="right"></param>
///<returns></returns>
publicstaticbool EvaluateStringExpression(string left, string right, OperandEnum o)
    {
switch (o)
        {
caseOperandEnum.EQ:
return left == right;
caseOperandEnum.NEQ:
return left != right;
default:
returnfalse;
        }
    }

///<summary>
/// Resolves a data type
///</summary>
///<param name="value"></param>
///<returns></returns>
publicstaticDataTypeEnum ResolveDataType(string value)
    {
if (Regex.IsMatch(value, @"^\d+$"))
        {
returnDataTypeEnum.INTEGER;
        }
elseif (Regex.IsMatch(value, "^\\d+(\\.\\d+)?$"))
        {
returnDataTypeEnum.FLOAT;
        }
elseif (value.ToLower() == "true" || value.ToLower() == "false")
```

```
        {
returnDataTypeEnum.BOOLEAN;
        }
else
        {
returnDataTypeEnum.STRING;
        }
      }

      #endregion

      #region Public Methods

///<summary>
/// Evaluates the dynamic expression
///</summary>
///<returns></returns>
publicbool Evaluate()
      {
if (this.LeftData != null&&this.RightData != null)
        {
if (this.LeftData.DataType == this.RightData.DataType)
          {
try
            {
switch (this.LeftData.DataType)
              {
caseDataTypeEnum.INTEGER:
return EvaluateIntegerExpression(Convert.ToInt32(this.LeftData.Value),
Convert.ToInt32(this.RightData.Value), this.Operand);
caseDataTypeEnum.BOOLEAN:
return EvaluateBooleanExpression(Convert.ToBoolean(this.LeftData.Value),
Convert.ToBoolean(this.RightData.Value), this.Operand);
caseDataTypeEnum.FLOAT:
return EvaluateFloatExpression(Convert.ToDecimal(this.LeftData.Value),
Convert.ToDecimal(this.RightData.Value), this.Operand);
caseDataTypeEnum.STRING:
return EvaluateStringExpression(this.LeftData.Value.ToString(), this.RightData.Value.ToString(),
this.Operand);
              }
            }
catch { }
          }
        }
returnfalse;
      }

///<summary>
```

```
/// Resolves an operator by it's string representation
///</summary>
///<param name="operand"></param>
///<returns></returns>
publicOperandEnum GetOperator(string operand)
      {
switch (operand)
        {
case"==":
returnOperandEnum.EQ;
case"!=":
returnOperandEnum.NEQ;
case">":
returnOperandEnum.GT;
case">=":
returnOperandEnum.GTEQ;
case"<":
returnOperandEnum.LT;
case"<=":
returnOperandEnum.LTEQ;
default:
returnOperandEnum.EQ;
        }
      }

      #endregion
   }
}


using System.Collections.Generic;
using System.Text;
using System.Xml;
using OOWF.Lib.Entities.Definition;

namespace OOWF.Lib.Shared.Utils
{
///<summary>
/// Parses data field and resolves their values
///</summary>
publicclassDataFieldsParser
   {
      #region Public Properties

///<summary>
/// The datafield raw xml
///</summary>
publicstring DataFieldsXML { get; set; }
```

```
///<summary>
/// The collection of parsed datafields
///</summary>
publicList<DataField> DataFields { get; set; }

      #endregion

      #region Constructor

///<summary>
/// Default Constructor
///</summary>
public DataFieldsParser()
      {
this.DataFields = newList<DataField>();
      }

///<summary>
/// Overloaded constructor
///</summary>
///<param name="xml"></param>
public DataFieldsParser(string xml)
         : this()
      {
this.DataFieldsXML = xml;
      }

      #endregion

      #region Public Methods

///<summary>
/// Returns a datafield by name
///</summary>
///<param name="fieldName"></param>
///<returns></returns>
publicList<DataField> Parse(string xml)
      {
this.DataFields.Clear();
XmlDocument doc = newXmlDocument();
         doc.LoadXml(xml);
XmlNodeList nodes = doc.SelectNodes("DataFields/DataField");
if (nodes != null)
         {
foreach (XmlNode node in nodes)
           {
DataField d = newDataField();
```

```
                d.Id = XMLUtils.GetNodeAttribute(node, "Id");
                d.Name = XMLUtils.GetNodeAttribute(node, "Name");
                d.DisplayName = XMLUtils.GetNodeAttribute(node, "DisplayName");
                d.DataType = d.GetDataType(XMLUtils.GetNodeAttribute(node, "Type"));
                d.Value = XMLUtils.GetNodeAttribute(node, "Value");
this.DataFields.Add(d);
            }
        }
returnthis.DataFields;
    }

publicstring GetValue(string key)
    {
if (this.DataFields != null)
        {
foreach (DataField d inthis.DataFields)
            {
if (d.Name.Trim().ToLower() == key.Trim().ToLower())
                {
return d.Value.ToString();
                }
            }
        }
returnstring.Empty;
    }

///<summary>
/// creates the xml for the datafields
///</summary>
///<param name="d"></param>
publicstring GetXML()
    {
this.DataFieldsXML = string.Empty;
StringBuilder sb = newStringBuilder();

XmlWriterSettings writerSettings = newXmlWriterSettings();
        writerSettings.Indent = true;
        writerSettings.Encoding = newUTF8Encoding();

using (XmlWriter writer = XmlWriter.Create(sb, writerSettings))
        {
            writer.WriteStartDocument();
            writer.WriteStartElement("DataFields");

foreach (DataField d inthis.DataFields)
            {
                writer.WriteStartElement("DataField");
                writer.WriteAttributeString("Id", d.Id);
```

```
                writer.WriteAttributeString("Name", d.Name);
                writer.WriteAttributeString("DisplayName", d.DisplayName);
                writer.WriteAttributeString("Type", d.DataType.ToString());
                writer.WriteAttributeString("Value", d.Value.ToString());
                writer.WriteEndElement();
            }

            writer.WriteEndElement();
            writer.Flush();
            writer.Close();
        }

this.DataFieldsXML = sb.ToString();

returnthis.DataFieldsXML;
      }

      #endregion
   }
}

using OOWF.Lib.Shared.Enums;

namespace OOWF.Lib.Shared.Utils
{
///<summary>
/// Translates the shared enum's
///</summary>
publicclassEnumTranslator
   {
      #region Public Static Methods

///<summary>
/// Gets a status enum from text
///</summary>
///<param name="status"></param>
///<returns></returns>
publicstaticStatusEnum GetStatus(string status)
      {
switch (status.ToLower())
         {
case ("active"):
returnStatusEnum.ACTIVE;
case ("completed"):
returnStatusEnum.COMPLETED;
case ("cancelled"):
returnStatusEnum.CANCELLED;
default:
```

```
returnStatusEnum.NONE;
        }
      }

      #endregion
   }
}


using System;
using System.IO;

namespace OOWF.Lib.Shared.Utils
{
publicclassLogger
   {
privatestaticreadonlystring filePath = @"C:\WorkflowLog.txt";

      #region Static Methods

///<summary>
/// Logs to the the applications logfile
///</summary>
///<param name="message"></param>
publicstaticvoid Log(string message, System.Exception ex)
      {
StreamWriter logFile = newStreamWriter(filePath, true);
         logFile.WriteLine("{0}: {1}", DateTime.Now, message);
if (ex != null)
         {
            logFile.WriteLine(ex.ToString() + Environment.NewLine + "<----------------------------------------------
------------------------------------------>" + Environment.NewLine);
         }
         logFile.Close();
      }

      #endregion
   }
}

using System.Xml;
using OOWF.Lib.Entities.Base;
using OOWF.Lib.Entities.Definition;
using OOWF.Lib.Shared.Enums;

namespace OOWF.Lib.Shared.Utils
{
publicclassProcDefXMLParser
```

```
  {
    #region Instance Data

privateProcess _procDef;
privateXmlDocument _xmlDoc;
privateXmlNode _rootNode;
privateXmlNamespaceManager _nsManager;

    #endregion

    #region Constructors

///<summary>
/// Default Constructor
///</summary>
public ProcDefXMLParser()
    {
      _xmlDoc = newXmlDocument();
      _nsManager = newXmlNamespaceManager(_xmlDoc.NameTable);
      _nsManager.AddNamespace("xpdl2", @"http://www.wfmc.org/2008/XPDL2.1");
      _nsManager.AddNamespace("xpdExt", @"http://www.tibco.com/XPD/xpdExtension1.0.0");
      _nsManager.AddNamespace("email", @"http://www.tibco.com/XPD/email1.0.0");
      _procDef = newProcess();
    }

///<summary>
/// Overloaded Constructor
///</summary>
///<param name="filePath"></param>
public ProcDefXMLParser(string xmlDefinition) : this()
    {
      _xmlDoc.LoadXml(xmlDefinition);
      _rootNode = _xmlDoc.DocumentElement;
      _procDef.XMLDefinition = _xmlDoc.InnerXml;
    }

    #endregion

    #region Public Methods

///<summary>
/// Parses the process definition xml and returns a process object
///</summary>
///<returns></returns>
publicProcess Parse()
    {
// Parse the package header
this.ParsePackageHeader();
```

```
// Parse the Redefinable header
this.ParseRedefinableHeader();

// Parse the Participants
this.ParseParticipants();

// Parse the workflow process
this.ParseWorkflowProcess();

// Parse the data fields
this.ParseDataFields();

// Parse the Activities
this.ParseActivities();

// Parse the transitions
this.ParseTransitions();

returnthis._procDef;
      }

      #endregion

      #region Private Methods

///<summary>
/// Fills the details of an email task
///</summary>
///<param name="e"></param>
///<param name="definitionNode"></param>
privatevoid FillEmailTaskDefinition(EmailTask e, XmlNode taskNode)
      {
XmlNode emailNode = taskNode.SelectSingleNode("email:Email", _nsManager);
if (emailNode != null)
         {
XmlNode emailDefNode = emailNode.SelectSingleNode("email:Definition", _nsManager);
if (emailDefNode != null)
            {
// To
XmlNode innerNode = emailDefNode.SelectSingleNode("email:To", _nsManager);
               e.To = XMLUtils.GetNodeInnerText(innerNode);

// Subject
               innerNode = emailDefNode.SelectSingleNode("email:Subject", _nsManager);
               e.Subject = XMLUtils.GetNodeInnerText(innerNode);
            }
```

```
// Body
XmlNode bodyNode = emailNode.SelectSingleNode("email:Body", _nsManager);
if (bodyNode != null)
            {
               e.Body = XMLUtils.GetNodeInnerText(bodyNode);
            }
         }
      }

///<summary>
/// Parses the Activities from the process definition
///</summary>
///<param name="node"></param>
privatevoid ParseActivities()
      {
XmlNode headerNode =
this._rootNode.SelectSingleNode(@"xpdl2:WorkflowProcesses/xpdl2:WorkflowProcess/xpdl2:Activities"
, _nsManager);
if (headerNode != null)
         {
foreach (XmlNode activityNode in headerNode.ChildNodes)
            {
Activity activity = null;
// Determine the Activity Type
XmlNode innerNode = activityNode.FirstChild;

string nodeName = innerNode.LocalName.ToLower();

if (nodeName == "event")
               {
// It is an event activity
                  activity = newEventActivity();

// Deterimine the event type
XmlNode eventNode = innerNode.FirstChild;
                  nodeName = eventNode.LocalName.ToLower();
if (nodeName == "startevent")
                  {
                     ((EventActivity)activity).EventType = EventTypeEnum.START;
                  }
elseif (nodeName == "endevent")
                  {
                     ((EventActivity)activity).EventType = EventTypeEnum.END;
                  }
else
                  {
                     ((EventActivity)activity).EventType = EventTypeEnum.INTERMEDIATE;
                  }
```

```
            }
elseif (nodeName == "implementation")
            {
XmlNode taskNode = innerNode.FirstChild.FirstChild;
                nodeName = taskNode.LocalName.ToLower();
if (nodeName == "taskmanual")
                {
// It is a manual task
                   activity = newManualTask();
                }
else
                {
// Must be a service task so determine the type
string serviceType = XMLUtils.GetNodeAttribute(taskNode, "xpdExt:ImplementationType").ToLower();
if (serviceType == "e-mail")
                   {
                      activity = newEmailTask();
this.FillEmailTaskDefinition((EmailTask)activity, taskNode);
                   }
                }
            }
else
            {
// Must be a route activity
                activity = newRouteActivity(); // Only supporting exclusive gateway
            }

// Get the performers
            innerNode = activityNode.SelectSingleNode("xpdl2:Performers", _nsManager);
if (innerNode != null)
            {
// Parse the activity performers
this.ParseActivityPerformers(activity, innerNode);
            }

// Add the activity to the collection
if (activity != null)
            {
this.SetBaseAttributes(activityNode, activity);
this._procDef.Activites.Add(activity);
this._procDef.ActivitiesHt.Add(activity.Id, activity);
this._procDef.ActivityNames.Add(activity.Name, activity.Id);
            }
         }
      }

   }
```

```
///<summary>
/// Parses the activity performers
///</summary>
///<param name="a"></param>
///<param name="node"></param>
privatevoid ParseActivityPerformers(Activity a, XmlNode node)
    {
foreach (XmlNode childNode in node.ChildNodes)
        {
Participant p = this._procDef.GetParticipantById(XMLUtils.GetNodeInnerText(childNode));
if (p != null)
            {
               a.Participants.Add(p);
            }
        }
    }

///<summary>
/// Parses the formal data field elements
///</summary>
privatevoid ParseDataFields()
    {
XmlNode headerNode =
this._rootNode.SelectSingleNode(@"xpdl2:WorkflowProcesses/xpdl2:WorkflowProcess/xpdl2:DataField
s", _nsManager);
if (headerNode != null)
        {
foreach (XmlNode childNode in headerNode.ChildNodes)
            {
DataField d = newDataField();

               SetBaseAttributes(childNode, d);

// Data Type
XmlNode innerNode = childNode.SelectSingleNode(@"xpdl2:DataType/xpdl2:BasicType", _nsManager);
               d.DataType = d.GetDataType(XMLUtils.GetNodeAttribute(innerNode, "Type"));

// Initial value
               innerNode = childNode.SelectSingleNode(@"xpdExt:InitialValues/xpdExt:Value",
_nsManager);
               d.Value = XMLUtils.GetNodeInnerText(innerNode);

               _procDef.DataFields.Add(d);
            }
        }
    }

///<summary>
```

```
/// Parses the Package Header Elements
///</summary>
privatevoid ParsePackageHeader()
    {
XmlNode headerNode = this._rootNode.SelectSingleNode("xpdl2:PackageHeader", _nsManager);
if (headerNode != null)
        {
// Created Date
XmlNode childNode = headerNode.SelectSingleNode("xpdl2:Created", _nsManager);
this._procDef.CreatedDate = XMLUtils.GetNodeInnerText(childNode);

// Description
            childNode = headerNode.SelectSingleNode("xpdl2:Description", _nsManager);
this._procDef.Description = XMLUtils.GetNodeInnerText(childNode);
        }
    }

///<summary>
/// Parses the workflow participants
///</summary>
privatevoid ParseParticipants()
    {
XmlNode headerNode = this._rootNode.SelectSingleNode("xpdl2:Participants", _nsManager);
if (headerNode != null)
        {
foreach (XmlNode childNode in headerNode.ChildNodes)
            {
Participant p = newParticipant();

                SetBaseAttributes(childNode, p);
                p.Type = ParticipantTypeEnum.HUMAN;

                _procDef.Participants.Add(p);
            }
        }
    }

///<summary>
/// Parses the Redefinable Header Elements
///</summary>
privatevoid ParseRedefinableHeader()
    {
XmlNode headerNode = this._rootNode.SelectSingleNode("xpdl2:RedefinableHeader", _nsManager);
if (headerNode != null)
        {
// Author
XmlNode childNode = headerNode.SelectSingleNode("xpdl2:Author", _nsManager);
this._procDef.Author = XMLUtils.GetNodeInnerText(childNode);
```

```
// Version
            childNode = headerNode.SelectSingleNode("xpdl2:Version", _nsManager);
this._procDef.version = XMLUtils.GetNodeInnerText(childNode);
          }
       }

///<summary>
/// Parses the transition elements
///</summary>
privatevoid ParseTransitions()
       {
XmlNode headerNode =
this._rootNode.SelectSingleNode(@"xpdl2:WorkflowProcesses/xpdl2:WorkflowProcess/xpdl2:Transition
s", _nsManager);
if (headerNode != null)
          {
foreach (XmlNode childNode in headerNode.ChildNodes)
             {
Transition t = newTransition();

                SetBaseAttributes(childNode, t);

// From
                t.From = XMLUtils.GetNodeAttribute(childNode, "From");

// To
                t.To = XMLUtils.GetNodeAttribute(childNode, "To");

// Transition Type
XmlNode innerNode = childNode.SelectSingleNode("xpdl2:Condition", _nsManager);
if (innerNode != null)
                {
// Condition Type
string conditionType = XMLUtils.GetNodeAttribute(innerNode, "Type");
if (conditionType == "CONDITION")
                   {
                      t.TransitionType = TransitionTypeEnum.CONDITION;

// Expression
                      innerNode = childNode.SelectSingleNode(@"xpdl2:Condition/xpdl2:Expression",
_nsManager);
                      t.Expression = XMLUtils.GetNodeInnerText(innerNode);
                   }
else
                   {
                      t.TransitionType = TransitionTypeEnum.OTHERWISE;
                   }
```

```
            }

// Add the transition info to the activities
Activity fromActivity = null;
Activity toActivity = null;
if (!string.IsNullOrEmpty(t.From))
            {
if (this._procDef.ActivitiesHt.ContainsKey(t.From))
              {
                fromActivity = (Activity)this._procDef.ActivitiesHt[t.From];
              }
            }
if (!string.IsNullOrEmpty(t.To))
            {
if (this._procDef.ActivitiesHt.ContainsKey(t.To))
              {
                toActivity = (Activity)this._procDef.ActivitiesHt[t.To];
              }
            }
if (fromActivity != null&& toActivity != null)
            {
              toActivity.IncomingActivityIds.Add(fromActivity.Id);
              toActivity.IncomingTransitions.Add(t);
              fromActivity.OutgoingActivityIds.Add(toActivity.Id);
              fromActivity.OutgoingTransistions.Add(t);
            }

            _procDef.Transitions.Add(t);
          }
        }
      }

///<summary>
/// Parses the main workflow elements
///</summary>
privatevoid ParseWorkflowProcess()
      {
XmlNode headerNode =
this._rootNode.SelectSingleNode(@"xpdl2:WorkflowProcesses/xpdl2:WorkflowProcess", _nsManager);
if (headerNode != null)
        {
          SetBaseAttributes(headerNode, _procDef);
        }
      }

///<summary>
/// Sets the base attributes for a base entity
///</summary>
```

```
///<param name="node"></param>
///<param name="e"></param>
privatevoid SetBaseAttributes(XmlNode node, BaseEntity e)
      {
         e.Id = XMLUtils.GetNodeAttribute(node, "Id");
         e.Name = XMLUtils.GetNodeAttribute(node, "Name");
         e.DisplayName = XMLUtils.GetNodeAttribute(node, "xpdExt:DisplayName");
      }

      #endregion
   }
}


using System.Xml;

namespace OOWF.Lib.Shared.Utils
{
publicclassXMLUtils
   {
      #region Public Static Methods

///<summary>
/// Gets a specified attribute for an xml node
///</summary>
///<param name="node"></param>
///<param name="attribute"></param>
///<returns></returns>
publicstaticstring GetNodeAttribute(XmlNode node, string attribute)
      {
if (node != null)
         {
if (node.Attributes[attribute] != null)
            {
return node.Attributes[attribute].Value;
            }
         }
returnstring.Empty;
      }

///<summary>
/// Gets the inner text of an xml node
///</summary>
///<param name="node"></param>
///<returns></returns>
publicstaticstring GetNodeInnerText(XmlNode node)
      {
if (node != null)
```

```
        {
if (!string.IsNullOrEmpty(node.InnerText))
          {
return node.InnerText;
          }
        }
returnstring.Empty;
      }

      #endregion
  }
}



namespace OOWF.Lib.Shared.Enums
{
publicenumBehaviorTypeEnum
  {
    PARALLEL,
    INCLUSIVE,
    EXCLUSIVE,
    COMPLEX,
  }
}


namespace OOWF.Lib.Shared.Enums
{
publicenumDataTypeEnum
  {
    BOOLEAN,
    FLOAT,
    INTEGER,
    STRING,
    UNDEFINED,
  }
}


namespace OOWF.Lib.Shared.Enums
{
publicenumEventTypeEnum
  {
    START,
    INTERMEDIATE,
    END,
  }
```

```
}


namespace OOWF.Lib.Shared.Enums
{
publicenumOperandEnum
  {
    EQ,
    NEQ,
    GT,
    LT,
    GTEQ,
    LTEQ,
  }
}


namespace OOWF.Lib.Shared.Enums
{
publicenumParticipantTypeEnum
  {
    HUMAN,
    SYSTEM,
  }
}


namespace OOWF.Lib.Shared.Enums
{
publicenumRestrictionTypeEnum
  {
    JOIN,
    SPLIT,
  }
}


namespace OOWF.Lib.Shared.Enums
{
publicenumStatusEnum
  {
    NONE,
    READY,
    ACTIVE,
    CANCELLED,
    ABORTING,
    ABORTED,
    COMPLETING,
```

```
        COMPLETED,
    }
}


namespace OOWF.Lib.Shared.Enums
{
publicenumTransitionTypeEnum
    {
        NONE = 0,
        CONDITION = 1,
        OTHERWISE = 2,
    }
}
```

Appendix E: Aspect Oriented Source Code

```
using System;
using AOWF.Lib.Entities.Base;
using AOWF.Lib.Shared.Utils;
using PostSharp.Aspects;
using PostSharp.Extensibility;

namespace AOWF.Lib.Aspects
{
///<summary>
/// This aspect is used to log whenever an acitivty is executed
///</summary>
   [Serializable]
publicsealedclassActivityLoggingAspect : OnMethodBoundaryAspect
   {
public ActivityLoggingAspect()
      {
this.AttributeInheritance = MulticastInheritance.Multicast;
      }

// Invoked at runtime if an execption occurs
publicoverridevoid OnException(MethodExecutionArgs args)
      {
Logger.Log(((RuntimeEntity)args.Instance).DisplayName + " has failed to execute.", args.Exception);
      }

// Invoked at runtime before that target method is invoked.
publicoverridevoid OnEntry(MethodExecutionArgs args)
      {
Logger.Log(((RuntimeEntity)args.Instance).DisplayName + " has started executing.", null);
      }

// Invoked at runtime after the target method is invoked (in a finally block).
publicoverridevoid OnSuccess(MethodExecutionArgs args)
      {
Logger.Log(((RuntimeEntity)args.Instance).DisplayName + " has finished executing.", null);
      }
   }
}

using System;
using AOWF.Lib.Shared.Enums;
using AOWF.Lib.Aspects;
using PostSharp.Extensibility;
```

```
namespace AOWF.Lib.Entities.Base
{
///<summary>
/// Represents a runtime workflow object
///</summary>
publicabstractclassRuntimeEntity : BaseEntity
   {
      #region Public Properties

///<summary>
/// The process instance this activity belongs to
///</summary>
publicstring ProcessInstanceId { get; set; }

///<summary>
/// The process status
///</summary>
publicStatusEnum Status { get; set; }

///<summary>
/// The process Created date
///</summary>
publicDateTime CreatedDate { get; set; }

///<summary>
/// The process completed date
///</summary>
publicDateTime? CompletedDate { get; set; }

      #endregion

      #region Public Methods

///<summary>
/// Executes this activity
///</summary>
      [Aspects.ActivityLoggingAspect(AttributeInheritance = MulticastInheritance.Multicast)]
publicabstractvoid Execute();

///<summary>
/// Saves the instance to the database
///</summary>
publicabstractvoid Save();

      #endregion
   }
}
```