

# Ant based interactive tools for workflow management

Ahmet Sayar\*

\*Department of Computer Engineering, Kocaeli University  
Kocaeli, Turkey  
ahmet.sayar@kocaeli.edu.tr

**Abstract**—The Apache Ant presents certain characteristics that seem to make it suitable as a workflow language and engine. Features such as native dependency structured build targets make it easy to write declarative dependency-based workflows.

We propose a framework enabling Ant build files (and its specifications) to be used as workflow scripts. Features like being semi structured and having standard specification make it easy to create, parse and render the workflow scripts. Sometimes workflow scripts need to be used repeatedly with a little change, framework enables reusable scripts to be created, stored and edited for later use. This allows sharing and dynamic modifications of workflows between multiple participants. The framework is composed of web-based interactive admin and user tools enabling creation and invocation of workflow templates and corresponding scripts.

**Keywords** – workflow, ant, process, interactive and distributed

## I. INTRODUCTION

Ant is a build tool in the Java community, providing platform independence and immediate integration into the newly adopted systems. It is open source, written in Java, and freely distributed by [1]. Ant supports sequential and parallel execution containers that allow subtasks to be executed in sequence or in parallel, respectively. Ant provides a flexible mechanism to express script dependencies in a project build process. Ant tasks and dependencies are expressed in Extensible Markup Language (XML) [2].

Process management is very crucial in large scale science and business applications. Some processes might need to involve some other processes to fulfill their goals. These processes are called dependable processes and the sub-processes should run in coordination and in a synchronized way. Dependability and orderings among the sub-processes are defined by a user or a main process, in a form of a document defining orderings or hard-coded in the application. The documents defining orderings and dependability of tasks are basically named as *workflow scripts*.

Some people have more experience and have more knowledge for some specific jobs and others might need their help to create workflow script to run applications. First group of people is called *admin*, and second group

of people is called *users*. Users are actual application users. They convert template build workflow script files into actual workflow scripts (*instances*). These reusable templates help users to be able to use the applications appropriately. System provides users with the forms (*templates*) created with the admin-given parameters. Workflow scripts (*instances*) are created automatically by the system with the user-given parameters. Users also can see their outputs on the interactive screen after the job is done.

Sometimes workflow scripts need to be used repeatedly with a little change. Rewriting everything from the scratch and running them are time consuming. Instead of doing repeating parts, users need to update just changing parts. Updating and editing on previously created workflow definitions are done by storing/fetching them to/from nonvolatile storages. Storages are located on local server's file system.

Some jobs are too complicated to run for an application user. It can be seen more clearly if the job requires chained and orchestrated processes. In that case, user needs to understand what the job does or what the application is for. We have created admin and user tools enabling creation and invocation of Apache Ant build templates and corresponding instances remotely through graphical user interface. The proposed architecture gives the admin a control capability over the running parameters of the workflow instances created by the application users. During the creation of workflow instances from the script templates, some parameters can be set by the user but others can not. These settings are done by the admin at the Ant build template creation stage. In order to run an application through Ant workflow script instance, application users open their interfaces on the browser, select one of the workflow script templates, fill it out, run it, and get the output on his/her screen.

This paper is structured as follows. The first section gives background information about Ant and present related works using Ant for workflow definitions. Section III mentions about general framework of the proposed architecture. Section IV is about the conclusion and future work.

## II. RELEATED WORK

There are lots of workflow projects in both commercial (Inconcert [4]) and academic (WSFL [5]) research area. Among these, GridAnt [6] is a pioneer project using Ant for the process pipelining. It's approach motivated many research projects [7] including us to use Ant as workflow engine. This section gives brief explanation about the usage of Ant in workflow and the differences from other approaches. At the end of the section, our work is compared with GridAnt project.

The Apache Ant project presents certain characteristics that seem to make it suitable as a workflow language and engine. Features such as native dependency structured build targets make it easy to write declarative dependency based workflows. One can consider workflows that can be described by an acyclic graph, where nodes of the graph represent a task to be performed and edges represent dependencies between tasks. This is harder to represent with a scripting language without a substantial additional framework behind it, but it is not at all difficult to represent with a tool like Ant [8].

Ant is designed around the concept of targets and tasks. Targets exist only as top-level entities and represents jobs. Dependencies between targets are specified using target names as handles. Targets in Ant are executed sequentially, without the possibility to exploit the parallelism inherent in a dependency-based specification. Targets are composed of tasks which are generally executed sequentially. Parallel execution of tasks can be achieved with the <parallel> task. It executes its nested tasks in parallel, synchronously. Globally scoped immutable properties can be used to provide value abstractions. Conditional execution is achieved at the target level based on property values.

GridAnt uses the workflow engine available with Apache Ant and develops a Grid [9] [10] workflow vocabulary on top of it. GridAnt is an XML/Java-based tool for representing and executing the flow of control around scientific computational codes and web services. It also provides a set of Grid tasks to be used within the Ant framework. Among these are authentication, file-transfer, and job-execution. It provides the ability to use features from Ant such as the XML specifications of tasks, the control flow specification through conditional, sequential, and parallel constructs, and the availability of a workflow processing engine. As GridAnt provides an abstract definition of elementary tasks performed in Grids it is possible to use the same application specification on a variety of Grid hosting environments.

GridAnt extends Ant functionalities and capabilities to decrease its deficiencies in using as a workflow tool, but we use Ant as it is and propose an intermediary architecture enabling creation of Ant build tools as workflow scripts. We also provide application users with the tools to run distributed chained processes remotely in accordance with the predefined workflows. We propose general tasks and focus on the creation of script based workflow templates interactively.

## III. ARCHITECTURE

Ant is designed around the concept of targets and tasks. Targets exist only as top-level entities. Dependencies between targets are specified using target names as handles. Targets in Ant are executed sequentially, without the possibility to exploit the parallelism inherent in a dependency-based specification. Targets are composed of tasks that are generally executed sequentially. In the chain each process is defined in tag "target" and flow orderings are defined by attribute "depends". For example, if job A depends on job B, then job B should precede job A and target defining job B should have attribute "depends=job A". In the current implementation errors are not handled.

The proposed framework is composed of two major parts (see Figure 1). One is for creating workflow templates (Admin Tools) and another is for creating final workflow script instances (User Tools). All the templates and instances are kept in the servers' local file systems as different Ant build files. Users and admin interact with the system through a browser. These two servers' roles are illustrated in two different colors. These roles are explained in the following sections, III.A and III.B.

In Figure 1, templates for the predefined workflows are listed as T-1, T-2, T-3 and T-4. These are created by admin users. When the users want to run any workflow, they need to select one of these templates and enter their own parameters for the job to execute and return the results. T1 I-1, T1 I-2 and T1 I-3 are three workflow instances created from template T-1 by the actual users. Administrators (admin users) and actual users both interact with the system through browsers.

Workflow templates as shown in Figure 1 are stored in *repository* implemented as local file system on the server. They are actually XML based Ant build files. Users interact (add, edit, delete) with them through visualization components of the system. Apache Ant's standard specification for the Ant build files is used as metadata

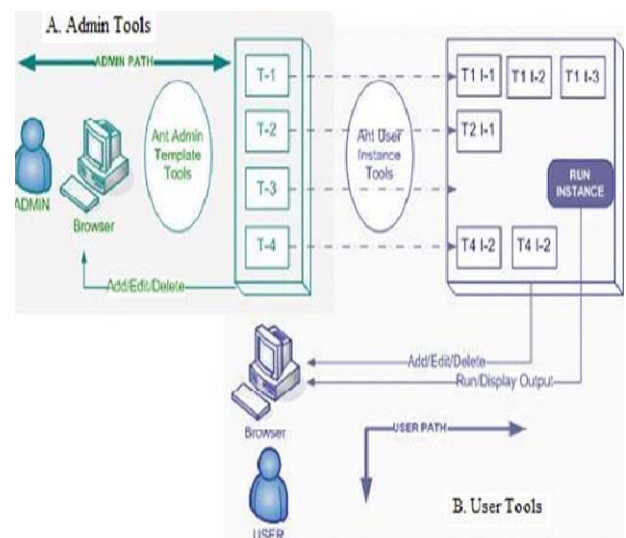


Figure 1. General architecture covering admin and user tools

for the templates to be created and edited. It also enables us to create interactive visualization components over the templates.

The aim in designing a workflow repository was to dynamically include workflow templates or provide the ability to modify the templates while a workflow is in progress. Remote access to the repository is also an important consideration in order to utilize the components of the workflow system in a collaborative environment by providing remote workflow template storage and access to distributed group members.

#### A. Admin Tools

Admin is provided three capabilities. These are adding, editing and deleting template files. Any update or change is seen by any others instantly. Root element of the templates is element “project”. Project element might have more than one target elements. Each target element represents an independent task or job. If any task is related to any other task in order, this relation is expressed by using “depends” attributes of the target tag elements.

A template might be an instance in case of that admin does not give any option for users to enter, and the entire template is filled out and all the elements are assigned by the admin. If the admin wants to give ability for users to define any tag value, then that tag value is set to specific value such as “ON” in the template file. All the options and restrictions on template files and indirectly workflow instances are defined by the admin users. As an example, see the SShEXEC page of the project in Figure 2. In that sample, the admin frees the users to enter their Host names but job can be run only by username “asayar”. For the other attributes you can have a look at the Apache Ant web site [11].

All previously created templates are kept in local file systems and displayed upon request. After making any change on any template admin can save it with another name or rewrite on the current file.

```
<project>
  <target name="ls" depends="echo">
    ....
  </target>
  <target name="echo">
    ....
  </target>
</project>
```

The possible job wrapping tags in Ant are scp, sshexec and exec. Sshexec is shown as a sample during creation of template file in Figure 2.

```
<sshexec host="whale.cs.india.edu"
  Username="asayar"
  Password="*****"
  Command="<exec>echo</exec>"
  Trust=""true"
  Failonerror="true"
  Append="false">
</sshexec>
```

Figure 2. A sample workflow template created with admin tools.

You can put this sshexec element after creating using Figure 1 into target named “echo” above. You can create task “ls” in the form of sshexec and put it in target “ls”. At the end you will get a completed template or an instance. This example gives basic information about how to create a workflow script using Apache Ant. See Figure 3 a template workflow script ready for creating a workflow instance by a user.

#### B. User Tools

Users create actual workflow script instances to run. Instances are created from templates. When user opens a browser, he gets the list of available templates that he is allowed to use to create instances. Currently, authentication and authorization are not implemented for the users to get available template lists. This will be the future work.

After getting the available template, user selects one and sets his own parameters and runs (please see Figure 3). Whenever user updates any value on the left hand side of the figure, workflow script instance is updated on the right hand side. After having finished their own parameters, users run the script and get the output on the right hand side of the Figure 3. Without restarting the server or the browser, users can rerun the script with minor changes again and again.

The value of the user-editable parameters and tag attributes are written in text boxes, and others are already assigned by the admin and not allowed to change. These authorization settings are defined by the administrators through admin tools presented in A.

#### IV. CONCLUSION AND FUTURE WORK

The proposed architecture is composed of an engine and a set of visualization components that provide dynamic views of the progress through the execution graph. Currently any change done by any user or admin can be seen by any other user or admin. We plan to put admin and user tools in a portal [12] environment by converting them to portlets. Through the portlet each application user and admin has their own file space and context variables.

XSL [3] and XSLT [3] are two technologies to convert an XML file into another XML or plain text file. We plan to use XSL for creating graphical user interfaces from the template files created by admin. Currently we do that by using Java Server Pages (JSP) pages in a tree structure. Since Ant has a standard schema, it will allow us to create a generic XSLT files to convert them into HTMLs or any other human readable text files. However, Ant schema is too large, creating XSLT file for conversion might not help for the performance issues but it is worth to try.

Due to its main targeted usage area, Ant lacks several aspects in supporting sophisticated workflow requirements. Workflows created by using Apache Ant

are not powerful enough for most distributed applications. Ant lacks the functionality to support workflow compositions. Ant provides mechanisms only to direct the flow of control. It lacks the infrastructure to support workflow composition allowing the output of one activity to become the input to another and defining the number of iteration for any process execution. In the future we are going to enhance the proposed architecture by eliminating these drawbacks through some extension to the XML-based Ant build files.

We also plan to test and apply our proposed admin and user interactive tools to real geophysics applications such as Pattern Informatics (PI) [13] and Virtual California (VC) [14].

#### REFERENCES

- [1] "The Apache Software Foundation," 2010; <http://www.apache.org/>.
- [2] J. Clark and S. DeRose, *XML Path Language (XPath) Version 1.0*, Specification REC-xpath-19991116 1999.
- [3] J. Clark, *XSL Transformations (XSLT)*, REC-xslt-19991116, W3C, 1999.
- [4] S.K. Sarin, "Object-Oriented Workflow Technology in InConcert," *Proc. 41st IEEE International Computer Conference (COMPCON)*, IEEE Computer Society, 1996, pp. 446.
- [5] F. Leymann, *Web Service Flow Language (WSFL)*, Standard Specification, IBM, 2001.
- [6] G.v.L. Kaizar Amin, Mihael Hategan, Nestor J. Zaluzec, Shawn Hampton, Albert Rossi, "GridAnt: A Client-Controllable Grid Work.ow System," *Proc. 37th Annual Hawaii International Conference on System Sciences (HICSS)* IEEE Computer Society, 2004, pp. 70210c.
- [7] Z. XU, et al., "Ant algorithm-based task scheduling in grid computing," *Proc. Canadian Conference on Electrical and Computer Engineering* IEEE Computer Society, 2003, pp. 1107-1110
- [8] G. Fox and D. Gannon, "Workflow in Grid Systems," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, 2006, pp. 11; DOI 10.1002/cpe.1019.
- [9] F. Berman, et al., *Grid Computing: Making the Global Infrastructure a Reality*, John Wiley & Sons, 2003.
- [10] M. Atkinson, et al., "Web Service Grids: An Evolutionary Approach " *Concurrency & Computation: Practice&Experience*, vol. 17, no. Number 2-4, February/April 2005, 2005, pp. 377-389.
- [11] "Apache Ant," 2010; <http://ant.apache.org/>.
- [12] J. Novotny, et al., "GridSphere: An Advanced Portal Framework," *Proc. 30th EUROMICRO Conference*, 2004, pp. 412-419.
- [13] J.R. Holliday, et al., "A RELM Earthquake Forecast Based on Pattern Informatics," *Book A RELM Earthquake Forecast Based on Pattern Informatics*, Series A RELM Earthquake Forecast Based on Pattern Informatics, ed., Editor ed.^eds., 2005, pp.
- [14] K.F. Tiampo, et al., "Eigenpatterns in southern California seismicity," *Journal of Geophysical Research*, vol. 107, no. B12, 2002, pp. 2354; DOI 10.1029/2001JB000562.

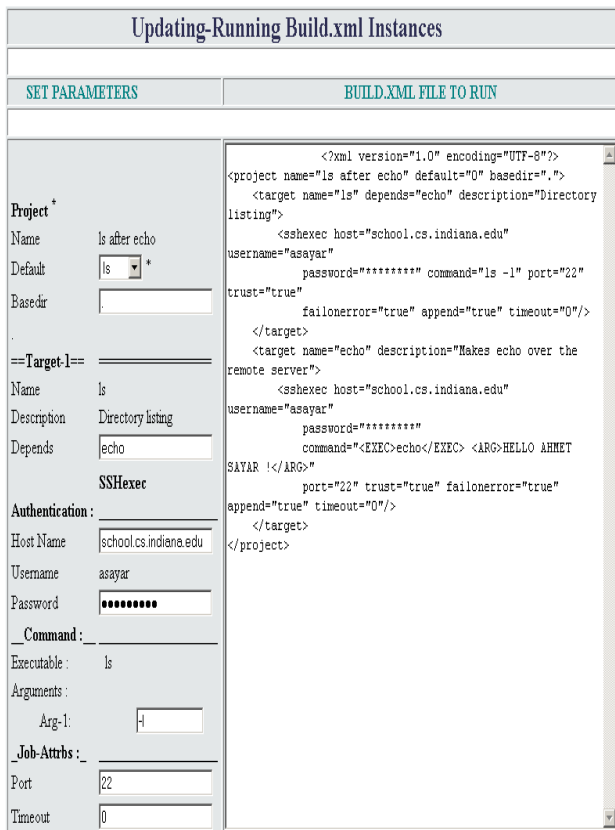


Figure 3. (User tool) A sample template is edited, and a workflow script is created.