OraAutomator Specification

Distributed Systems Services, EDS Bahrain

1 Revision History

Version	Date	Changes
1.0	March 26, 2007	Initial version - DRAFT

2 Introduction

This document describes the proposed *OroAutomator* system, intended to automate the installation and configuration processes of Oracle clients as well as front-end applications that use them. The system is modeled using Z, which is good at succinctly and unambiguously describing systems. The user is not expected to be proficient in the notation; prose and footnotes are included throughout the document. Please note that this model portrays the system at a particular level of abstraction, hence the reason why details that some readers may consider to be important have been left out.

3 Model

3.1 State Space Definition

[EXECUTION, FRONT_END, MIDDLEWARE]

EXECUTION_PROCESS == **P** EXECUTION

YES_NO ::= yes | no

The system has three basic types: executions, front-end applications and middleware applications. We don't care what they are, just as long as we can define variables of these types in our model. We define an *EXECUTION_PROCESS* as a power set of *EXECUTION*, meaning that if *EXECUTION* looks something like the following

{crowns, aries, step, airman, ora_dev45, …}

Then *EXECUTION_PROCESS* will look like the following

{{}, {crowns}, {crowns, step}, {crowns, step, aries}, …}

As you can see, this means that *EXECUTION_PROCESS* contains all the possible permutations of installable applications[1].

OraAutomator

$$\begin{array}{l} app : FRONT\_END \rightarrow MIDDLEWARE \\ install\_in\_progress : YES\_NO \\ exec\_process : \subseteq EXECUTION\_PROCESS \end{array}$$

[1] Note that the current assumption is that specifying the name of a front-end application alone in an execution process installs its accompanying middleware application, if applicable, as well.

Our state space says that our system, *OraAutomator*[2], has three variables. *app* is a function that maps front-end applications to middleware ones since every front-end application that we care about requires some middleware application. *install_in_progress* is a lock meant to ensure that the pre-installation, installation and post-installation tasks for a particular application (discussed later) are done atomically so that, say, an application doesn't perform its pre-installation tasks while another is performing its installation tasks. *exec_process* is a subset of the power set of all possible execution processes, mentioned earlier. The idea is that each time the system is executed, it sequentially goes through an execution process and then performs all the necessary operations for each one of its applications[3].

3.2 State Space Initialization

InitOraAutomator

OraAutomator

---

exec_process = {}
install_in_progress = no
app = {

$crowns \mapsto ora\_dev45,$
$aries \mapsto ora9,$
$step \mapsto ora\_dev6,$
$airman \mapsto ora9,$
$discoverer\_ora35 \mapsto <>,$
$discoverer\_ora40 \mapsto <>,$
$hermes\_admin \mapsto ora9,$
$easy \mapsto jinitiator,$

}

When the system is initialized, we define the set of applications and assume an empty *exec_process* set. A separate schema allows users to define the execution process. An empty range means that the application's front-end already includes the required connectivity that is usually achieved by the middleware.

[2] *a : b : variable a is of type b*
[3] Installation specifics can be worked out later

3.3 Operations

PreInstall

$\Delta\ OraAutomator$

$front\_end\_app? : FRONT\_END$

$middleware\_app? : MIDDLEWARE$

$cleanup\_higher\_versions : YES\_NO$

$front\_end\_app? = \langle\rangle \vee middleware\_app? = \langle\rangle$

$cleanup\_higher\_versions = yes \Leftrightarrow middleware\_app? \neq \langle\rangle$

$front\_end\_app? \in exec\_process \vee middleware\_app? \in exec\_process$

$install\_in\_progress = no$

$install\_in\_progress' = yes$

$exec\_process' = exec\_process$

$app' = app$

The PreInstall operation[4] takes care of all the tasks for a particular application that have to be done prior to installation. In the case of installing an older version of an Oracle client for example, it means removing all higher versions of the client on the local machine by cleaning up the registry and file system.

Our schema here says that PreInstall accepts two inputs: a front-end application and a middleware one and has a variable that determines whether or not it should go ahead and do a cleanup (for the case described earlier). The invariants that follow simply say that 1) the *install_in_progress* flag must be low before the operation starts and high after it exits, 2) the execution process and the set of applications remain unchanged and 3) both the *front_end_app* and *middleware_app* inputs may not be non-null, i.e. we can't pre-install a front-end application and a middleware one concurrently. The alternative of course would be to split PreInstall into two operations for each of the two types of applications.

INSTALL_TYPE ::= normal | custom

Install

$\Delta\ OraAutomator$

$front\_end\_app? : FRONT\_END$

$middleware\_app? : MIDDLEWARE$

$install\_type? : INSTALL\_TYPE$

$front\_end\_app? = \langle\rangle \vee middleware\_app? = \langle\rangle$

$install\_in\_progress = yes$

$exec\_process' = exec\_process$

$app' = app$

[4] Δ a : This operator modifies the state space a, a $\in$ b: a is an element of b, a $\Leftrightarrow$ b: a is true if and only if b is true, $\vee$ *: logical or (disjunction)*

Our Install operation takes in an extra input variable called *install_type* that determines how the application will be installed. Install also makes sure that the *install_in_progress* flag is set to high before it does anything.

PostInstall

$\Delta\ OraAutomator$

$front\_end\_app? : FRONT\_END$
$middleware\_app? : MIDDLEWARE$
$restore\_higher\_versions : YES\_NO$

$restore\_higher\_versions = yes \Leftrightarrow middleware\_app? \neq \langle\rangle$
$install\_in\_progress = yes$
$install\_in\_progress' = no$
$exec\_process' = exec\_process$
$app' = app$

PostInstall includes variables and invariants similar to the ones discussed earlier.

DefineExecutionProcess

$\Delta\ OraAutomator$

$exec\_set\_ordered? : \subseteq EXECUTION\_PROCESS$

$exec\_process' = exec\_set\_ordered?$
$install\_in\_progress' = install\_in\_progress$
$exec\_process' = exec\_process$
$app' = app$

DefineExecutionProcess simply sets our system's *exec_process* variable to the one defined by the user. Operations such cleaning up higher versions of, say, Oracle, and thereafter restoring them, configuring applications during and after installation are all considered, but remain implicit in this model.

GetMiddleware

$\Xi\ OraAutomator$

$front\_end\_app? : FRONT\_END$
$middleware\_app! : MIDDLEWARE$

$middleware\_app! = app(front\_end\_app?)$

GetMiddleware is used to retrieve the middleware application associated with a particular front-end application, during consistency checks for example.

4 Sample Sequence

The user simply has to supply the system with an execution process; the system then performs the necessary pre-installation, installation and post-installation tasks, including application-specific configurations and checks for each of the applications in that execution process[5].

5 Next Steps

Following a revision of this draft specification by the author and the approval of all stakeholders, the design and subsequent implementation of the system may be done by the author or one of the team members. It may be necessary to expand the model to include finer details such as the steps involved in checking for installed components, performing system-wide cleans, etc.

[5] Of course, the system has to get that information from somewhere, so the user will have to define application-specific information pertaining to installation and configuration procedures prior to executing the system.