ITIL Process Automation as a Test Bed for SOA

Michael Rulf
USinternetworking, an AT&T Company

You are equipped with a technical understanding of Web Services. You are a strong believer in the power of Service Oriented Architecture (SOA). Now you're eager to bring SOA to your enterprise. You want to get maximum benefit from SOA, so you propose to service-enable the key functions of your company's enterprise resource planning (ERP) and customer relationship management (CRM) applications and automate cross-application processes like order-to-cash.

Quickly, you realize that this means significant change to the organization. Unfortunately, most human beings and organizations resist change. Your CIO is doubtful. He doesn't want to spend any resources on anything that isn't absolutely necessary. He demands 100% proven return on investment. Functional users are limited to their functional silos. Having discussions about cross-application processes requires 20 people to be in one room and still nobody has ownership of the end-to-end process.

Don't be discouraged. There are many ways to get started with SOA. Rather than attacking the most complex cross-application business processes, you can, for example, apply SOA principles to traditional data integration challenges. You can also focus on hot topics such as Web self-service to demonstrate the power of Web Services. In this article, we'll show you another interesting approach to how you can get started with SOA in a less-daunting setting.

We'll show you how you can practice SOA and build up value step-by-step. The first is to service-enable small units of existing custom code. Second, we'll show how you can then reuse these Web Services as part of small process flows. Third, these smaller process flows become useful sub-processes of end-to-end business flows. That was your goal, right?

Let's reveal the secret: we believe IT operations are an interesting test bed for SOA, specifically for SOA-based process management. IT processes are what the CIO understands best. IT processes aren't as organizationally challenging as large end-to-end business processes. Yet, complex IT environments have created IT operations that are ready for process improvement. Today, system administrator salaries are the most prominent IT budget line item, and those highly paid resources are executing mundane, ad hoc, and unscalable tasks. It's no wonder then that ITIL (Information Technology Infrastructure Library), a widely accepted framework for IT Service Management, is today's "hot topic." Hence, it's likely your CIO will listen to your proposal to apply SOA to the IT operations challenge. The good news is that system administrators have written large amounts of custom code in the form of Perl code, Unix shell scripts, or Visual Basic (VB) scripts. These programmatic assets are ideal candidates to apply your skills of service-enabling IT assets.

USinternetworking, Inc. (USi), an AT&T company, made use of this favorable environment for its endeavor into SOA. Let's learn from USi's success. Concretely, we'll explain how you can quickly create a Web Service from an existing Perl script. Then, we'll reuse the service-enabled Perl script as part of a typical IT process and show how you can incorporate this IT process into a larger end-to-end business process. Finally, we'll dive into the world of IT processes to give you a better feel for the opportunity and how BPEL (Business Process Execution Language) is suited to this domain.

Service-Enabling a PERL Script

The first step in applying SOA in IT operations is to create a set of Web Services. On its quest to automate IT processes, USi made an important discovery: many of the tasks required already existed as Unix shell scripts, Perl code, and VB scripts. System administrators created such code components to simplify their job by automating repetitive tasks and generating the necessary documentation required for audit purposes. In the spirit of SOA, USi decided to reuse these pockets of custom code. By creating some basic "wrappers," these legacy code components were converted into Web Services. By standardizing this wrapper definition, it's possible to take disparate custom code created by a wide range of individuals and expose it as part of a SOA initiative with a common methodology for instantiation, security, and error reporting.

Let's take a look at how you can create such a wrapper for a legacy code component. If you come from the Unix/Linux world, you might be familiar with legacy scripts and code packages written using Perl. USi's Quadir Kareemullah demonstrates how easy it is to convert Perl code, in this case a script to create a Unix account for a given Linux server, into a Web Service. Listing 1 shows UserAdd.pm, the initial Perl script.

Listing 1: UserAdd.pm

```
#!/usr/local/bin/perl
#
# Author: Quadir Kareemullah USi
#
# Purpose: UNIX account creation module
#
package UserAdd;

use strict;
use warnings;

use Data::Dumper;

my $groupadd_cmd = "groupadd";
my $useradd_cmd = "useradd";
my $def_homedir = "/home";
my $def_shell = "/bin/bash";

sub useradd {
  my ($fullname, $username) = shift;

  my $result = { error => 1, status => "Could not run useradd() function" };

  my $cmd = "$groupadd_cmd $username 2>&1";
  my $out = `$cmd`;
```

```
  if($out =~ /exists/) {
    $result->{error} = 1;
    $result->{status} = "Could not add $username user, Group $username already exists.";
  }
  else {
    $cmd = qq($useradd_cmd -c "$fullname" -d $def_homedir/$username -s $def_shell -g $username $username
2>&1);
    $out = `$cmd`;
    if($out =~ /exists/) {
      $result->{error} = 2;
      $result->{status} .= "Could not add $username user, User $username already exists.";
    }
    else {
      $result->{error} = 0;
      $result->{status} = "User $username successfully added.";
    }
  }
}

1;
```

Let's create a wrapping package that will expose UserAdd.pm via SOAP (Simple Object Access Protocol). The "use UserAdd" statement tells the new module to use our legacy code in UserAdd.pm. The two SOAP lines that follow are what enable the module to be called as a Web Service. A "sub" section then follows for each function in the original Perl module. A separate wrapping module lets you add additional SOAP error handling to conform to a Web Services methodology as found in the "die" statement near the end of the script. This wrapping package, UserAddWS.pl, is shown in Listing 2.

Listing 2: UserAddWS.pl

```
#!/usr/local/bin/perl

package UserAddWS;

use strict;
use warnings;
use diagnostics;

use Data::Dumper;

use lib './modules';
use UserAdd;

use SOAP::Transport::HTTP;

SOAP::Transport::HTTP::CGI
-> dispatch_to('UserAddWS')
-> handle;

sub useradd {
  my $self = shift;

  my $result = UserAdd::useradd(@_);

  if($result->{error}) {
    die SOAP::Fault->faultcode( $result->{error} )
      ->faultstring( $result->{status} );
  }
```

```
 return $result->{status};
}

1;
```

The result is that the original code continues to exist without any change. A significant benefit is that no regression testing is required for adding the wrapper in this non-intrusive manner.

In addition, other features can be introduced to UserAddWS.pl to take care of security. Now that the code is exposed as a Web Service, traditional application security that restricts access to the functionality is no longer in effect. However, support for standards such as WS-Security (Web Services Security) to manage access at the wrapper and WSDL (Web Services Description Language) level can be added without impacting the original application code.

The final step is to create a WSDL definition for the new package so that it can be called by a BPEL process or via statically typed languages such as Java or VB. If you search the Comprehensive Perl Archive Network (www.cpan.org), you'll find Perl modules that help automate the creation of WSDL files. The WSDL for our example code, UserAddWS.wsdl, is shown in Listing 3.

Listing 3: UserAddWS.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions name="UserAddWS"
 targetNamespace="http://mdsxaaw05.usi.net/wsdl/UserAddWS.wsdl"
 xmlns="http://schemas.xmlsoap.org/wsdl/"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:tns="http://mdsxaaw05.usi.net/wsdl/UserAddWS.wsdl"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">


 <message name="UserAddRequest">
  <part name="fullname" type="xsd:string"/>
  <part name="username" type="xsd:string"/>
 </message>

 <message name="UserAddResponse">
  <part name="status" type="xsd:string"/>
 </message>

 <portType name="UserAddWS_PortType">

  <operation name="useradd">
   <input message="tns:UserAddRequest"/>
   <output message="tns:UserAddResponse"/>
  </operation>

 </portType>
```

```
 <binding name="UserAddWS_Binding" type="tns:UserAddWS_PortType">

  <soap:binding style="rpc"
   transport="http://schemas.xmlsoap.org/soap/http"/>

  <operation name="useradd">

   <soap:operation soapAction=""/>

   <input>
    <soap:body
     encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
     namespace="urn:UserAddWS"
     use="encoded"/>
   </input>

   <output>
    <soap:body
     encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
     namespace="urn:UserAddWS"
     use="encoded"/>
   </output>

  </operation>

 </binding>

 <service name="UserAddWS_Service">

  <documentation>WSDL File for UserAddWS</documentation>

  <port binding="tns:UserAddWS_Binding" name="UserAddWS_Port">
   <soap:address
    location="http://mdsxaaw05.usi.net/soap/UserAddWS.pl"/>
  </port>

 </service>

</definitions>
```

In this WSDL, you can see a number of components relating to the UserAddWS.pl package, including the:

- Namespace definitions
- Request and response message types for each function defined in the package
- Port definition with an operation entry for each function in the package tied to the associated message types
- Binding definition with operation entries to tie all of the definitions together
- Service definition that defines the URL necessary to call the UserAddWS.pl package

As you've seen, the new Perl wrapper "UserAddWS.pl" and the WSDL file "UserAddWS.wsdl" together enable you to convert the legacy Perl package

"UserAdd.pm" into a Web Service. Following this methodology, you can rapidly service-enable an entire library of scripts used for IT operations.

Oracle's Enterprise Manager system management offering serves as another example how to make use of this technique to service-enable applications relevant to IT operations. The Enterprise Manager Command Line Interface (EM CLI) lets you access functionality from text-based consoles (shells and command windows) with custom scripts such as Perl, SQL*Plus, OS shell, or Tcl.

Hence, by using EM CLI in a Perl script and then exposing the Perl script as a Web Service as illustrated earlier, you can create Web Services for system management operations like monitoring/managing targets, jobs, groups, blackouts, notifications, and alerts. Concretely, sample Web Services could include:

- IT infrastructure provisioning actions such as adding/deleting targets, submitting/deleting jobs, creating/deleting users
- Monitoring functions such as every day send an e-mail list of backup jobs that were still running after 6am
- Management actions such as every week write pertinent information about failed system management jobs to a file and then purge the system management job history.

Listing 4 illustrates an EM CLI example that can be used to provision a hardware server, using configuration properties from the input file.

Listing 4: emcli provision

```
emcli provision
    -image="Images/myimage"
    -network="Networks/networkprofile"
    -bootserver="booservername.us.oracle.com"
    -stageserver="stageserver.us.oracle.com:/private/share"
    -stgcredentials="joe"
    -schedule="type:immediate"
    -resettimeout="100"
    -target="mylabel"
    -input_file="config_properties:properties.txt"
    -assignment="provision mylabel"
```

Our "emcli provision" submits a job that will provision the image file located under the filepath "Images/myimage" on the hardware server target specified in the variable "mylabel." This system management job will run immediately with a reset timeout of 100 minutes. Image properties will be picked up from the file properties.txt overwriting default image properties. The server "stageserver.us.oracle.com:/private/share" will be used as the staging server and "/private/share" will serve as the staging storage for this task executed with "joe" as the username.

Leveraging the technique we described, you can now execute this hardware server provisioning task via a Web Service by launching this command in a Perl script and wrapping the Perl script as a Web Service.

The main advantage in using Perl or shell scripts as a basis for Web Services is that this technique can be very widely applied. Most platforms provide command-line utilities to interact with a multitude of programs, all of which are available to scripting languages. This technique also transcends the Unix world. For example, Microsoft provides administrative utilities under Windows for Exchange and other products via PowerShell. Using Perl, you can invoke any of these commands programmatically and, in turn, expose them as Web Services as illustrated.

Automating an IT Process That Uses the Service-Enabled PERL Script

Now, let's create some value for the organization around the Perl script that we've turned into a Web Service. First, we want to reduce manual intervention, especially by a specialist such as the system administrator, who should worry about keeping the servers up and running, not executing mundane tasks such as provisioning accounts. There's a shortage of skilled system administrators and, besides, any manual intervention is error-prone and comes with a time delay (because many system administrators work off account provisioning requests in batches). Second, auditability is key in IT operations. Large data centers need to conform to audit standards such as SAS 70 (Statement of Auditing Standards No. 70). Frankly, for an application service provider such as USi, this was a key driver for applying SOA to IT operations. USi must comply with a number of audit standards, such as SAS 70 Type II, as part of its own corporate policies and procedures. In addition, USi is also subject to audit reviews by its clients in support of their compliance initiatives for Sarbanes-Oxley, HIPAA, and other regulatory legislation.

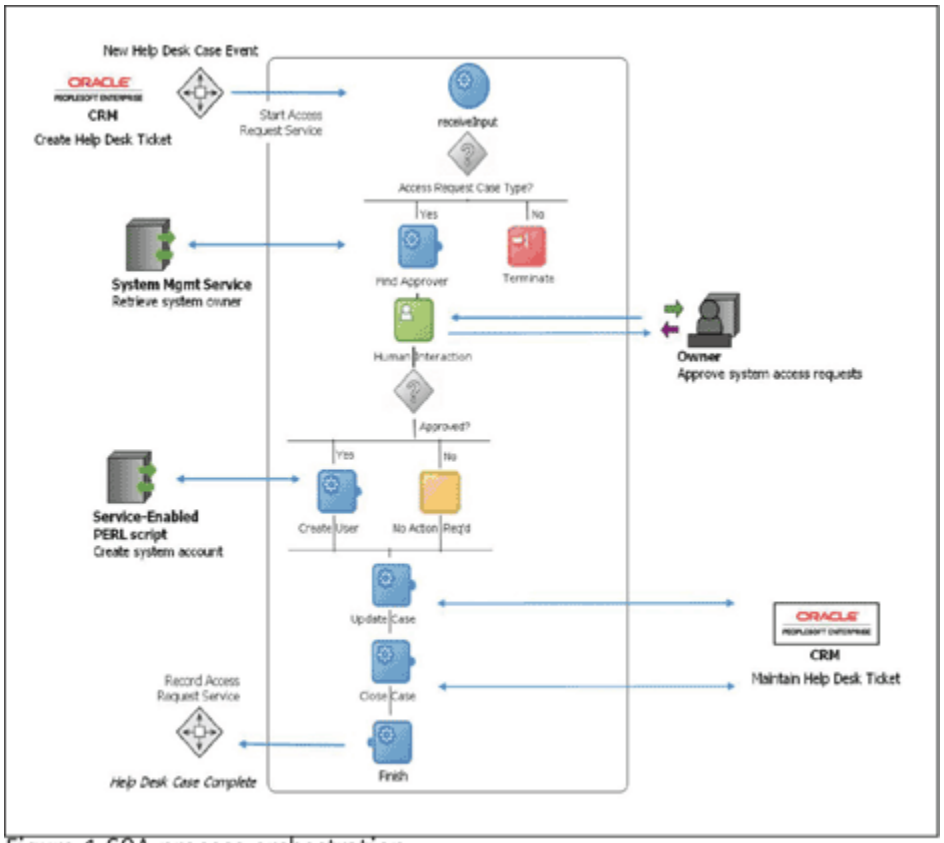

Figure 1 SOA process orchestration

Figure 1 illustrates how the service-enabled Perl script is embedded into an end-to-end IT operations process orchestrated by BPEL - the SOA process orchestration standard.

Besides the Perl script, two typical IT operations applications participate in this process. The first is an IT help desk system. USi uses PeopleSoft CRM as its help desk system. Your company might use help desk software from Remedy or Peregrine. Any help desk system has interfaces to interact with a BPEL orchestration engine. The second system is a system management application. For USi, this is a custom-developed application, but it could also be a system such as HP OpenView, IBM Tivoli, or Oracle Enterprise Manager, among others. Again, these systems either have ready-made Web Service interfaces or you can Web Service-enable their command-line interfaces.

The process kicks off with a request for a new Unix account on a specific server entered into the help desk system. It then queries the system management application to identify the server's owner. This owner is asked to approve the request in a human workflow step. Oracle BPEL Process Manager provides extensions to easily implement such human workflow steps. Once approved, the Perl script is called to execute the account provisioning request. Then, the help desk ticket is updated and finally closed.

This process automation produces some major benefits. First, the approval step is electronically documented, which meets SAS 70 Type II audit standards. Second, we limited any waiting time to the necessary human approval step. Every other activity, including the automatic lookup of the owner as well as the execution of the Perl script, is fully automated. The process can be initiated as a Web self-service request with the help desk system simply tracking the activity. Importantly, every BPEL process is again a Web Service. This means that this process can be immediately reused as part of larger business processes. In our case, a fitting context would be an HR employee onboarding process with the creation of a Unix account being one of many other steps. The advantages of repeatability, auditability, and the reduced execution times of the IT operations process are now inherited by the higher-level HR onboarding process, which is important to executive management. SOA has made a true difference to the business.

Your Opportunity to Apply SOA to ITIL Processes

The Unix account creation process was just a very simple example of an IT operations process where SOA can add value. We tried to exemplify how the combination of (1) service-enabling custom management scripts, (2) process orchestration including human approval steps, and (3) tying in typical applications used in IT operations, such as help desk and system management systems, can help streamline IT operations and even higher-level business processes.

To fully grasp the opportunity, you need to understand the current environment in IT operations. Over the last decade, IT environments have grown more and more heterogeneous and so complex. Separate subgroups in IT, such as networking, server, database, and applications management, often act as silo'd organizations using different help desk tools. Moreover, technology components, such as servers, networks, storage solutions, and applications, come with their own management point solutions - narrow in

focus and simply inadequate for delivering on the automation and control required. This results in IT operations teams implementing highly inefficient processes that involve significant manual effort. It therefore comes as no surprise that 70% of the IT budget is spent on maintenance, with administration headcount being the most prominent budget line item; 60%-80% of outages are due to human error; and highly paid system administrators are executing mundane, ad hoc, unscalable tasks. In brief: IT operations is ready for process improvement.

We are certainly not the first to point out this dilemma. IT managers are under increasing pressure to deliver on demanding service requests and to adapt to changing business needs. Processes in search of optimization include a broad spectrum of different tasks, such as:

- IT incident management (fault/problem remediation)
- IT problem management (root cause analysis)
- IT configuration management
- IT change management
- IT release management

IT organizations are realizing that they need enforceable and repeatable IT processes to improve the efficiency, effectiveness, and quality of their IT services (control); drive compliance with regulatory requirements such as Sarbanes-Oxley (compliance); and reliably support key business needs (alignment). Hence, they're looking to proven best practices as a means of gaining control. In this context, the ITIL IT Service Management process framework has become popular as guidance on how to tame the chaos that often thrives in modern data centers. Started in the mid-1980s by the United Kingdom's Office of Government Commerce, formerly known as the Central Computer and Telecommunications Agency, ITIL is now a mature, non-proprietary IT process framework that is industry- and technology-independent. ITIL has taken off over the last years, with an adoption rate of 45% expected in the United States in 2008. In Europe, where ITIL was originated, adoption is already ubiquitous.

Given the significant need for IT process improvement and broad consensus in the ITIL process framework, focus is then squarely on the technical enablement and automation of ITIL processes. BPEL, the SOA standard for process orchestration, provides a solution to implement ITIL process automation. BPEL was designed to leverage existing software functionality from multiple sources, such as ERP systems, service providers, and custom code via Web Service interfaces. It provides orchestration to enforce consistent execution of a defined process across these assets. Furthermore, in BPEL, processes are modular "building blocks" that can be flexibly assembled into larger end-to-end processes.

When USi embarked on IT process automation, it quickly realized that traditional workflow tools would be too restrictive. Today's IT operating environments require a complex network of supporting system and infrastructure components generating a large volume of "events" to continuously sort through. Traditional workflow tools tend to be very linear in terms of how they execute a process flow, with one step following the

other. They're not a good fit for the highly event-driven nature of IT management. Also, for USi, the dynamic nature of the approval process across many internal and client systems required an asynchronous solution. This in turn needed to be coupled with a rules engine for defining business logic to determine the necessary approvers at the time of execution. Oracle BPEL Process Manager provided USi with a platform to meet these requirements, including several added features that provide more value from an audit perspective. Each time a process flow executes, the BPEL orchestration engine provides a graphical view of the process flow. This enables end users to determine where a particular approval request currently stands in the overall business process. The BPEL engine also captures all of the task data, such as the approver and time of approval, for presentation to an auditor. And it provides revision control for the process definition to track when and how the process changed over time - invaluable functionality during audit reviews.

Perhaps, most important, USi learned to plan for growth from the outset when implementing Web Services and SOA technologies. Web Services have a way of quickly proliferating when you start automating discrete tasks. If you don't plan for this growth upfront by investing in a UDDI repository, or using other methods for tracking your toolbox of Web Services, you're destined to have multiple services with overlapping functionality. Reuse and repeatability is key to realizing the value of SOA. Using a UDDI repository promotes this by providing a way to locate existing Web Services in your toolkit so you can capitalize on the fruits of the discovery process you went through to find the scripts and automated tasks you converted to Web Services.

For you as an SOA architect or developer, IT operations are a worthwhile opportunity to shine with your SOA skill set. Most certainly, people will be impressed to see how you can service-enable existing Perl code, Unix shell scripts, or VB scripts, and connect them in conjunction with services provided by help desk and system management applications. More than that, you'll be able to demonstrate business value to your CIO and build momentum for a larger SOA initiative in your enterprise. If you want to go beyond the basics of BPEL, the "BPEL Cookbook" (http://oracle.com/technology/pub/articles/bpel_cookbook/index.html) is a great advanced resource to learn directly from your peers.

About the Authors:

Markus Zirn
Markus Zirn is a senior director of product management for Oracle Fusion Middleware. He heads the Strategic Customer Program, where he works with Oracle's most innovative middleware customers. Recently, he produced the "SOA Best Practices-The BPEL Cookbook" series on Oracle Technology Network. He has practical experience designing and optimizing business processes - conducting multiple business process re-engineering projects while a consultant with Booz Allen Hamilton. He holds a master's degree in electrical engineering from the University of Karlsruhe, Germany; the University of Southampton, U.K.; and ESIEE, France.

Michael Rulf
Michael Rulf is the vice president of advanced engineering at USinternetworking, Inc., an AT&T Company. At USi, he provides strategic direction and management of all product engineering initiatives with a primary focus on Oracle?s ERP products such as Siebel, Oracle E-Business Suite, and PeopleSoft. This includes setting strategic direction and managing development of various SOA initiatives encompassing Oracle ERP products, Microsoft business productivity products such as SharePoint, and e-commerce solutions such as IBM WebSphere Commerce. Recently, he designed and implemented an Oracle-based identity management system that provisions users to a range of applications using an SOA architecture, resulting in significant industry recognition.

Rajiv Taori
Rajiv Taori is the director of product management for Oracle Enterprise Manager, where he is responsible for its application management offerings. He is ITIL certified and has consulted and worked with companies on ITIL projects for more than five years. He has also worked as a management consultant with McKinsey & Company where he advised clients on their business, product, and technology strategies. He holds an MBA from the Wharton School of the University of Pennsylvania and a bachelor's degree in electrical engineering from the Indian Institute of Technology (IIT), Bombay.