At Your Service:

TestTrack Pro's Role in a Service-Oriented Architecture

It is a rare developer who has not heard the phrase "service-oriented architecture" bandied about. Just what is a service-oriented architecture (SOA)? And, what exactly does this have to do with my issue management system, TestTrack Pro? In this paper we'll explain why service-oriented architectures have come into vogue, the technologies used to implement them, and how TestTrack Pro fits into the big picture.

What Is SOA?

The concept of a service-oriented architecture is not new. Applications have long been built to provide services to other applications. What distinguishes SOAs from other architectures is that services are loosely-coupled through the use of platform-neutral Web interfaces.

Loose coupling is a good thing. It frees the client from needing to know what language the service is coded in or what platform the service runs on. The client communicates with the service through a well-defined interface, and then leaves it up to the service implementation to perform the necessary processing.

Compare this to services based on DCOM or other component-based architecture interfaces. In these cases the client must have significant insight into how the service was implemented—and the client must be coded just right to avoid technical snafus.

Why SOA?

The popularity of Web services-based SOAs is primarily due to their inherent reusability, interoperability, flexibility, and cost efficiency.

Reusability

No developer wants to reinvent the wheel, but many do, over and over. The holy grail of software development has long been the reuse of existing functionality. Enterprise developers faced with a hodgepodge of computing environments have often run into this challenge.

Solutions developed in different enterprises, even in different departments in the same enterprise, are often coded in languages with different programming interfaces and protocols. While an application may contain business logic that could, in theory at least, be reused across the enterprise, the harsh reality of incompatible architectures results in unrealized potential.

In a Web services-based SOA the only thing the client needs to know about is the interface to a service. Implementation details are hidden, so application integration is much simpler.

Interoperability

Web services-based SOAs are designed to be interoperable. The interoperability lets clients and services communicate with each other no matter what platform they run on.

Platform neutrality and standards-based protocols and technologies ensure that Web services-based SOAs are accessible from both within and outside the enterprise (with appropriate security precautions).

Tightly-coupled SOAs, in contrast, are interoperable only with clients that are built on the same technology base as the service. They also have trouble spanning corporate firewalls and other security measures that hinder their functionality.

Flexibility

The simplicity of Web services-based SOAs results in more flexible software systems. Since the inner workings of a Web service are shielded from the client, developers can freely change the implementation. In a tightly-coupled architecture,

the different components of an application are tightly bound to each other, sharing semantics, libraries, and often sharing states. This implementation of SOA is anything but flexible.

Cost Efficiency

New technologies become popular when they make financial sense. Web services-based SOAs require only a modest effort to implement and even less effort to make use of in an application. Tightly-integrated SOAs are generally not cost efficient. They are costly to build, maintain, and extend, because changes in one component often requires changes in other components.

Web Services Primer

Web services are the technology underlying most contemporary service-oriented architectures. They are based on widely accepted and used W3C standards such as XML, SOAP, and WSDL. It is these standards that form the basis of TestTrack Pro's Web service functionality.

XML

The eXtensible Markup Language (XML) has become the de facto standard for describing data to be exchanged on the Web. With XML, data can be exchanged between incompatible systems. Since XML data is stored in plain text format, XML provides a software- and hardware-independent way of sharing data.

XML involves the use of tags that describe the contents of a document. An XML tag identifies the information in a document, and also the structure of that information.

For example, the following XML markup describes the structure of a TestTrack Pro phone number field.

```
<element name="PhoneNumber" type="ttns:PhoneNumber"/>
<complexType name="PhoneNumber">
  <sequence>
    <element name="numbertype" type="xsd:string"/>
    <element name="phonenumber" type="xsd:string"/>
  </sequence>
</complexType>
```

The plain text nature of XML makes it easy to see that a phone number in TestTrack Pro is a compound field that consists of two string elements, `numbertype` and `phonenumber`.

SOAP

SOAP (Simple Object Access Protocol) is a simple XML-based protocol that lets applications exchange information over HTTP. SOAP avoids the compatibility and security problems common to RPC-based (Remote Procedure Call) communication protocols like DCOM and CORBA. SOAP also provides a way for applications running on different operating systems, with different technologies and programming languages, to communicate.

WSDL

WSDL (Web Services Description Language) is an XML-based language for describing and accessing Web services. A WSDL document is written in XML that specifies the location of the Web service and the operations (or methods) the service exposes. For example, the following WSDL describes the TestTrack Pro Web service that deletes a defect.

```
<message name="deleteDefectRequest">
  <part name="cookie" type="xsd:long"/>
  <part name="defectNumber" type="xsd:long"/>
  <part name="summary" type="xsd:string"/>
</message>
```

Writing applications to take advantage of TestTrack Pro's Web services takes a surprisingly small amount of work. Many programming environments now understand WSDL and automatically generate classes to handle the XML serialization/deserialization and network communication.

TestTrack Pro as a Service

Now that we've covered the basics of service-oriented architectures and how they're implemented using Web services, it's time to delve into the way TestTrack fits into this new world.

Sure, you could continue using TestTrack Pro just as it comes out of the box. After all, TestTrack Pro's capable Web and rich client interfaces are probably why you purchased the tool in the first place. But don't just stop there; TestTrack Pro is also built to be used as a service to other applications. The TestTrack Pro Server concurrently supports access from the TestTrack Pro Web and rich clients, and also from applications that call the server's SOAP-based API.

The flexibility offered with TestTrack Pro's SOAP API opens the door to many interesting applications. For example, you can create a simple application to extract and report on defect information in new ways, or you can leverage TestTrack Pro's strong workflow and issue management capabilities to create new classes of applications.

TestTrack Pro's SOAP API lets you extract, modify, add, and combine TestTrack Pro data in ways that go far beyond what is possible using the provided Web and rich client interfaces.

Here's a short list of the ways other users are putting TestTrack Pro's SOAP API to use.

- View all assigned open defects from multiple TestTrack Pro databases in a WebSphere Portal application.
- Automate creation of an Excel spread sheet that reports recently closed and current open defects.
- Create a help desk Web page to allow entry of service requests into an IT database.
- Review active user accounts and disable users who are no longer working at the company.
- Weekly email notification of open records with completion dates in the coming week.
- Email-based signoff process for approval of sensitive requests in the IT help desk database.
- Import new records directly from an Excel spreadsheet.
- Auto-escalation of defects that meet specific criteria (such as those that have been open a month or more).
- Integrate TestTrack Pro with Microsoft Outlook so that defects are automatically created as Outlook tasks.
- Create a Web page to monitorlicense usage.
- Promote files in Surround SCM based on specific defects that have been fixed in TestTrack Pro.
- Manage the Human Resources review process via a Web application.

Easy Applications

Creating a TestTrack Pro SOAP client application is remarkably simple. The first step is to set up the TestTrack Pro SDK. The SDK contains everything you need to get started, including three SOAP client licenses (three concurrent connections) and exhaustive documentation of the API's data types and operations. You'll also need a Web server (Apache, IIS, Netscape, etc.) that is supported by the TestTrack Pro Server.

You can use just about any programming language to create a TestTrack Pro SOAP client application, including Perl, Java, C#, and Visual Basic .NET. Keep in mind that development moves faster if the programming environment has support for Web services (as is now common).

To illustrate the ease of creating a TestTrack Pro SOAP client application, we'll use the example of a C# application that escalates defects when they pass specific criteria. In this example, a defect will be escalated if it is "Open," has a priority lower than "Immediate," and is at least one month old. When a defect passes these criteria, the application will change the defect's priority to "Immediate."

Since our example is coded in C#, we'll assume that Microsoft Visual Studio .NET is the IDE used. Now here's where the benefit of using a Web services-aware programming environment such as Visual Studio .NET is apparent. Instead of having the developer hand code a TestTrack Pro SOAP interface class, Visual Studio .NET simply requests a WSDL description of the available TestTrack Pro services. It then processes the WSDL and generates a C# class, ttsoapcgi, that encapsulates the functionality of TestTrack Pro's SOAP API.

Get Coding

Opening a connection to the TestTrack Pro server couldn't be easier. You just need to create an instance of the ttsoapcgi class and logon to the database. It's really as simple as this:

```
// setup the connection to TestTrack Pro and
// logon to the database
soapServer = new ttsoapcgi();
soapServer.DatabaseLogon(database, username, password);
```

Once the application is logged on to the TestTrack Pro database, it's ready to query for a list of defects. We'll use the getRecordListForTable method for this purpose:

```
// return an array of defects from the current database
soapServer.getRecordListForTable(cookie, "Defect", "",
columnsToReturn);
```

The argument columnsToReturn is a simple array that specifies the columns to be returned. The getRecordListForTable method returns a CRecordListSoap object that contains the desired fields from each defect in the database.

Now that we've obtained a list of defects (that was easy), we simply process them in turn. The code for doing so requires walking through the list, inspecting the properties of each defect, and updating the priority if appropriate. As there is nothing specific to SOAP in most of this code we'll just show what happens when a defect is escalated.

```
//escalate the defect and update the record in TestTrack Pro
defect.priority = "Immediate";
soapServer.saveDefect(cookie, defect)
```

When the application is complete processing, it's advisable to close the connection to TestTrack Pro so another application can use it. This is accomplished in a single line of code:

```
// disconnect from the TestTrack Pro database
soapServer.DatabaseLogoff(cookie);
```

This example shows just how easy it is to use TestTrack Pro's SOAP interface. Of course, in real life the application must also deal with error checking and other exceptions (as should any application). Even so, it is quite likely you can set up TestTrack Pro for the first time and write a simple SOAP application all within an hour or two.

The Hub of Activity

If you're intent on moving towards a service-oriented architecture, you should seriously consider putting TestTrack Pro's Web services interface to use. You'll find that TestTrack Pro has many more uses than one would first consider. TestTrack Pro offers the reusability, interoperability, flexibility, and cost efficiency to be the hub of many service-oriented activities.

Seapine Software,Inc.

5412 Courseview Dr.,
Suite 200
Mason, OH 45040

Toll free: (888) 683-6456
Email: info@seapine.com
Web: www.seapine.com