XML APIs Testing Using Advance Data Driven Techniques

(ADDT)

Shakil Ahmad

August 15, 2003

Table of Contents

1. INTRODUCTION

The goal of this paper is to provide an overview of the testing methodology and approach developed by the Convergys Test Automation team for XML APIs testing. The emphasis is how to test XML APIs using XML schemas, and the reusability of XML APIs scripts by using different data.

The main points of this paper include:

- A technical section describing the CAT tool infrastructure
- How to use Advanced Data Driven Techniques (ADDT) in order make automated script data more flexible
- How to create template from XML Schema
- How to create XML scripts from template
- How to execute XML test scripts
- How ADDT improves testing, reliability, and quality

2. TEST AUTOMATION

2.1. Automation Methodology

The Convergys Test Automation Team has developed an Advance Data Driven Techniques (ADDT) methodology by using the Test Automation Engine, which is used to test XML APIs based on the input provided by tables. This technique is used to test our highly complex PC based Billing applications. This approach has been successfully implemented in all Development and Testing Organizations in Convergys for XML APIs regression, and new XML APIs feature testing. For the past three years it has been proven that Automation Testing can improve the reliability and the quality of our software products, and reduce man-hours. Automation testing also helps us to detect faults in earlier stages of software release cycles, which reduces the bug fixing cost dramatically. This technique is generic and can be used for all XML APIs.

2.2. Automated Tool

No 3rd party tool is required to test XML APIs. Convergys Auto Tester (CAT) will provide the front-end to create and run XML APIs automated test scripts in conjunction with the Automation Engine by using the Advanced Data Driven Techniques (ADDT).

2.2.1. Data Driven Testing

Data driven testing is when the automated test cases read the test data from an external source, such as a file/table, rather than having the values hard coded into their scripts. To make tests data driven, the user must write them to take parameters. These parameters define the way in which the test functions, allowing the user to test a variety of scenarios with the same automated test scripts by just changing the data.

The real value of automated testing only comes when the user begins to use data driven automated tests, especially the ADDT due to the increased data control it allows, as well as its flexibility

2.3. XML

The Extensible Markup Language (XML) is a simple, very flexible text format used to exchange a wide variety of data. It was approved by the World Wide Web Consortium (W3C) in February 1998. Since it enables businesses and their computer systems to communicate more easily, XML is the foundation for a whole new way of communicating across the Internet. XML creates documents that are well structured; as a result all languages based on XML are also well structured. This means that the data in XML is more easily used than that in non-XML-based documents.

2.4. DTD (Document Type Definition)

A DTD can be used to define the legal building blocks of an XML document.

2.5. XML Schema

XML Schemas allow machines to carry out rules made by the user. They provide a means for defining the structure, content, and semantics of an XML document. XML Schema is an XML based alternative to DTD (Document Type Definition). The XML Schema language is also referred to as XML Schema Definition (XSD). The purpose of an XML Schema is to define the legal building blocks of an XML document, just like a DTD.

An XML Schema:

- identifies elements that can appear in an XML document
- establishes attributes that can appear in an XML document
- is used to identify child elements
- groups the order of child elements
- characterizes the number of child elements
- identifies whether an element is empty or can include text
- provides data types for elements and attributes
- identifies default and fixed values for elements and attributes

2.5.1. XML Schema VS DTD

There are a number of reasons why XML Schema is better than DTD.

2.5.1.1. XML Schema has Support for Data Types

A strong point of XML Schemas is its support for various data types. Because of XML Schema's support for various data types it is simpler to describe permissible document content, validate the correctness of data, work with data from a database, define data facets (restrictions on data), define data patterns (data formats), and convert data between different data types.

2.5.1.2. XML Schemas use XML Syntax

Another strong point of XML Schemas is that they are written in XML. Because of this, the user doesn't have to learn another language. They can use an XML editor to edit their Schema files or an XML parser to parse their Schema files. It also allows users to manipulate their Schema with the XML DOM (Data Object Model), and transform their Schema with XSLT.

2.5.1.3. XML Schemas Secure Data Communication

When data is sent from a sender to a receiver it is essential that both parties have the same "expectations" about the content. With XML Schemas, the sender can describe the data in a way that the receiver will understand.

2.5.1.4. XML Schemas are Extensible

Because they are written in XML, XML Schemas are extensible, just like XML. This allows the user to reuse their Schema in other Schemas, create their own data types derived from standard types, and reference multiple schemas from the same document.

2.5.1.5. A Well-Formed XML document

A well-formed XML document is a document that conforms to the following XML syntax rules:

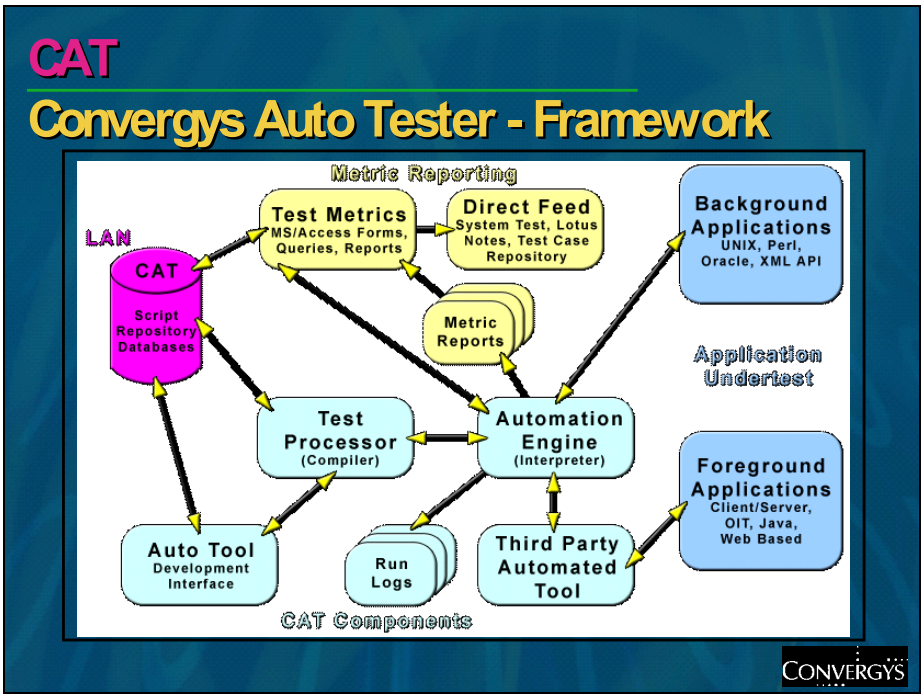
- Begins with the XML declaration
- Has one unique root element
- All start tags must match end-tags
- XML tags are case sensitive
- All elements must be closed and properly nested.
- All attribute values must be quoted
- XML entities must be used for special characters

3. CONVERGYS AUTO TESTER (CAT)

The Convergys Auto Tester (CAT) offers a powerful test automation tool set for client/server and web testing using ADDT, as well as a complete automation from test execution to report generation.

3.1.1. CAT – Architecture Framework

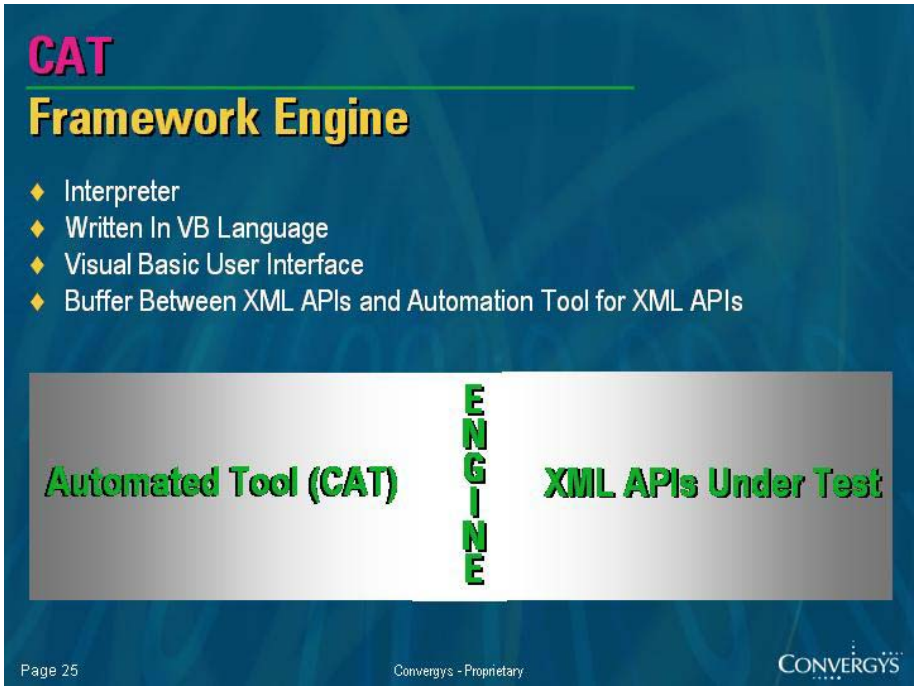
CAT architecture has been developed on a component basis and consists of a customized tool set and third party tools.



3.2. Convergys Auto Tester (CAT) Components

3.2.1. Engine

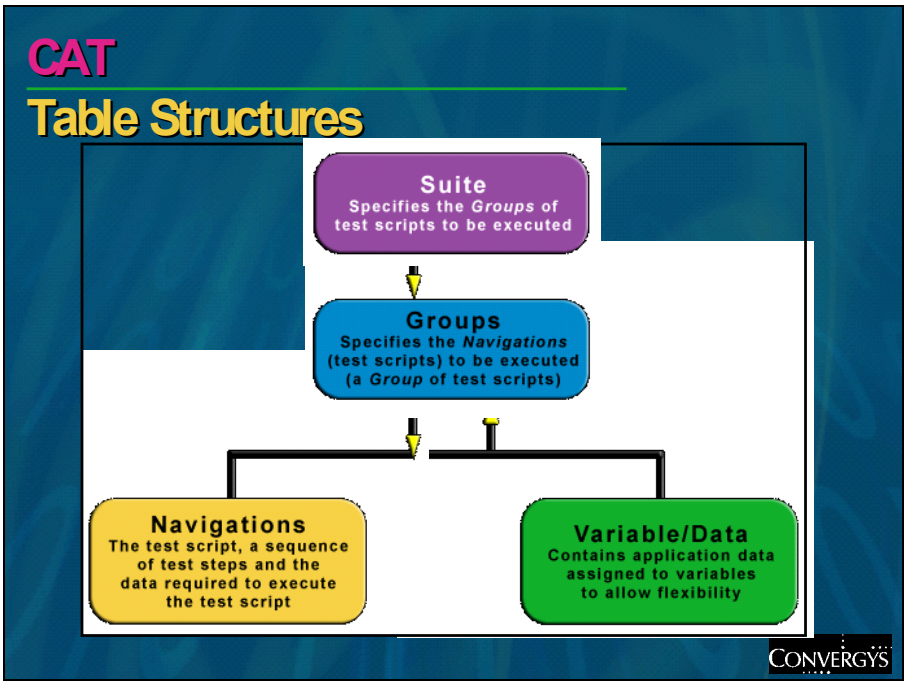
The **Engine** is a Translator; it consists of software modules, written in CAT Scripting Language (CSL), and Visual Basic programs, which provide a front-end interface for users. Third party tools, customized toolsets and error handling codes are embedded into the Engine for Foreground and Background processing including XML APIs. The Engine works as a buffer between the application under test and the automated tool.



3.2.2. Access Database

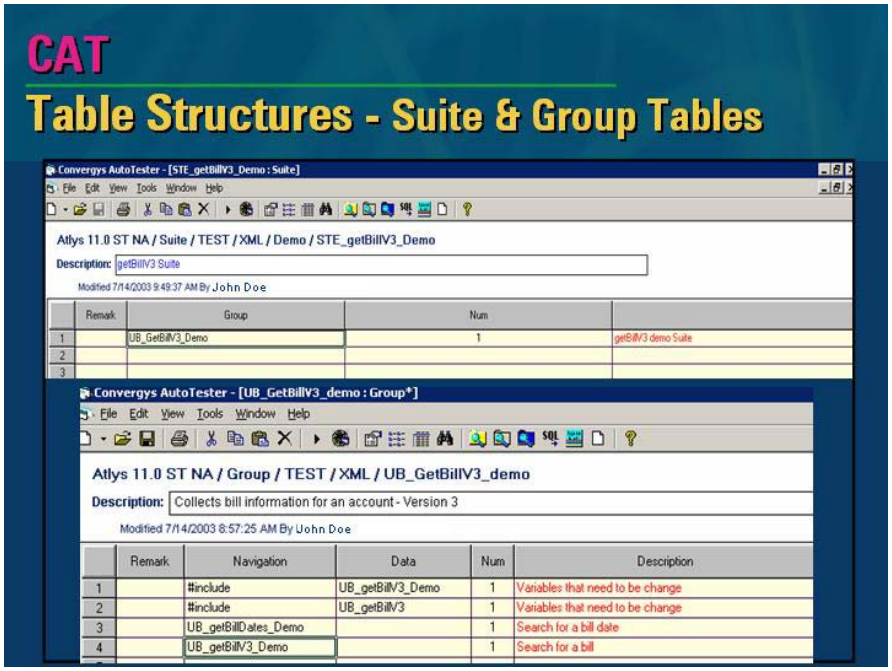
MS/Access database is used as a test case repository for automation scripts. The database is transparent to users, but they can access this database by using the Test Explorer. Users can update, modify, and delete their scripts. The database consists of four tables, Suite, Group, Navigation, and Data/Variable which are used for ADDT (Advance Data Driven Testing).

3.2.3. Table Structures



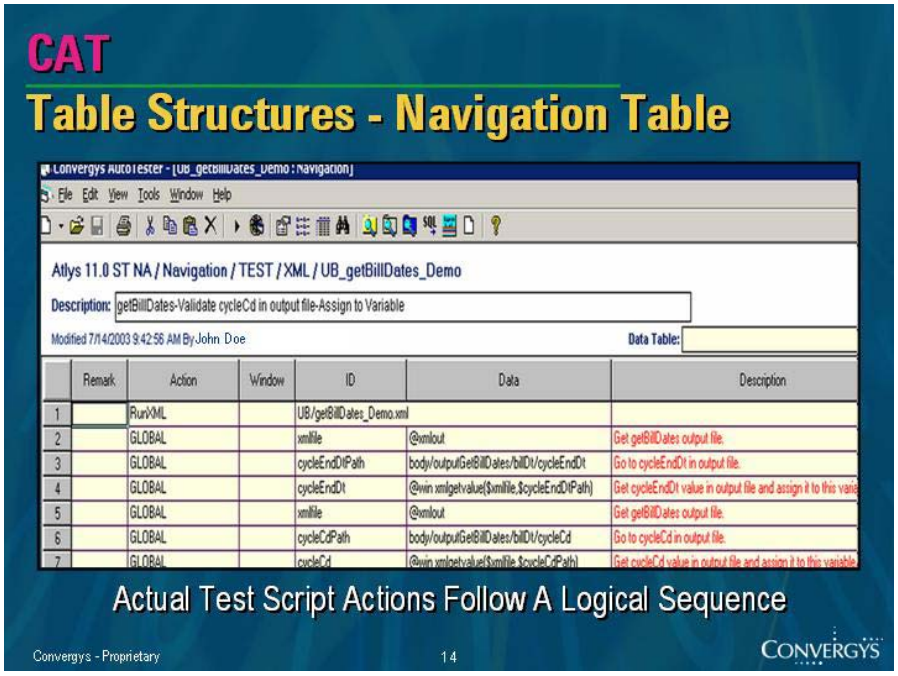
3.2.3.1. CAT – Suite and Group Tables

A **Suite** specifies the Groups of tests to be executed in a particular order. It is the highest organizational level for automated tests. A **Group** is a logical grouping of specific test cases (or Navigations) to be executed sequentially. Many Groups can make up one Suite.



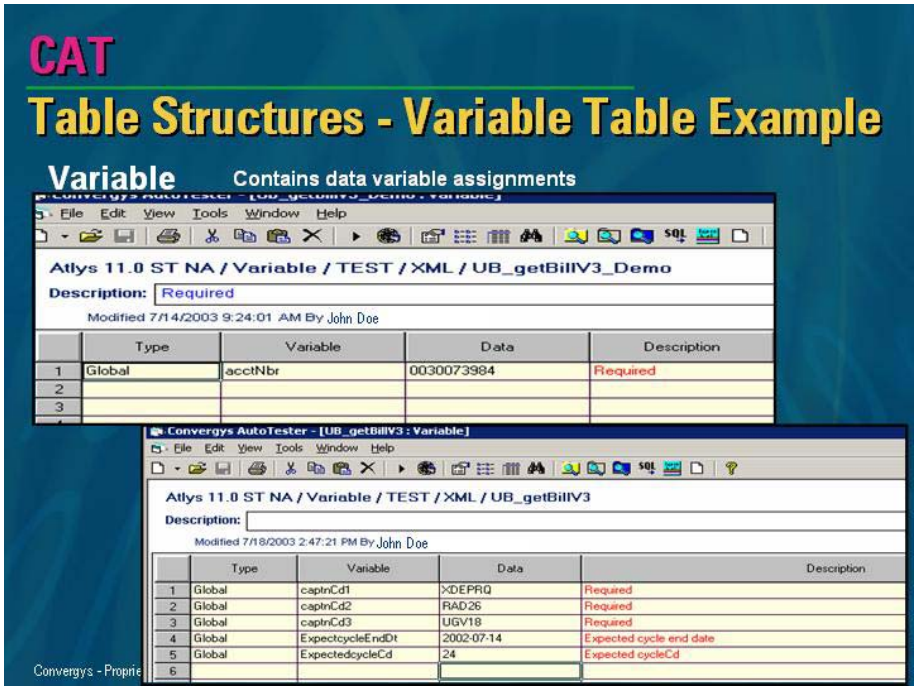
3.2.3.2. CAT – Navigation Tables

A **Navigation** is a sequence of steps or actions required to execute a test case. Navigation is considered a test case. Multiple Navigations make up a Group.



3.2.3.3. Variable Tables

A **Variable** table contains the specific data (names, date, address etc.). The variables can be use to specify the data or actual data can be hard coded. It will allow users to assign variable(s) to their data, for control and reusability.



3.3. ADDT Approach to Data

The Data/Variable table only allows the user to separate data from the actual automated script(s). In order to get more control of data and maximize Automation performance, the user needs additional functionality to make the data more flexible and reusable. To achieve these goals, the CAT tool has built-in functionality.

3.3.1. SQL Editor

More flexibility can be obtained by allowing users to create SQL scripts and save the scripts. Parameters can also be passed to the SQL scripts.

3.3.2. Variable Tables

Variable tables will allow users to assign variable(s) to their data, for control and reusability.

3.3.3. SQL/PERL Scripts

Execute SQL/PERL script(s) and re-assign to variable(s) for additional flexibility.

3.4. How to create XML test script template

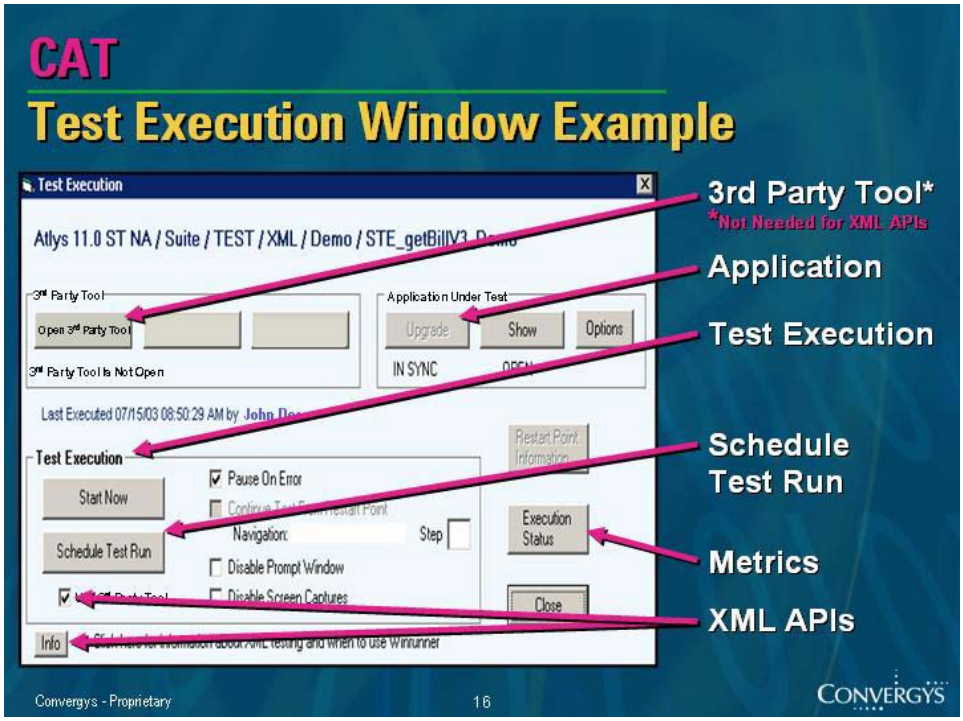
CAT tool has built-in functionality to create XML template(s) (The actual xml input file(s)).

Procedure to create XML test script using the CAT tool after the template has been created:

1. Use the built-in XML editor to create XML input file(s)
2. Create Navigation table(s)
3. Create Variable/Data table(s)
4. Create Group/Suite table(s)

3.4.1. How to execute XML test script(s)

Use the One-Stop CAT execution window to execute the XML test scripts. After execution, CAT tool will display Short Results Window(s) – synopsis of execution results, Details Log File(s) – detailed log of execution results including actual data information, Response Time Information, and Metrics Information.



4. HOW ADDT IMPROVES TESTING, RELIABILITY AND QUALITY

Using ADDT approach with Test Automation can greatly improve the testing reliability and quality:

- Reduced System Test duration
- Faster defect notification to development
- Improved testing coverage
- Improved client satisfaction
- Automatic proof of testing
- Test scripts can be developed before or parallel with the application development
- Easy to create and execute XML APIs script
- Data Driven, no “hard coded” data
- More data flexibility

5. BIBLIOGRAPHY

Connolly, Dan. XML: Principles, Tool, and Techniques. Travelers' Tales Inc., November 1997

Cover, Robin. Cover Pages: XML Articles and Papers. January-March 2000. Online. Available http://xml.coverpages.org/xmlPapers2000Q1.html, May 28, 2002

McKinnon, Linda, and Al McKinnon. XML in 60 Minutes a Day. John Wiley & Sons, May, 2003

O'Reilly & Associates. XML.com: XML From the Inside Out. Online. Available www.XML.com, July, 2003

Ray, Erik T. Learning XML. O'Reilly & Associates, February 2001

Van Der Vlist, Eric. XML Schema. O'Reilly & Associates, Inc., June, 2002