Case Study: Accelerating SAP Testing

SAP testing accelerators are a new trend from software testing vendors to introduce or facilitate automation testing efforts. SAP test accelerators are a prebuilt library of previously automated test cases representing SAP test transactions that can be customized or modified to meet a project’s specific and unique configuration settings. SAP test accelerators hold the promise of reducing the cycle time to automate SAP end-to-end processes (i.e., hire-to-retire, request-to-pay, etc.) while empowering the SAP project’s nontechnical members to assemble and execute automated test cases.

Although SAP testing accelerators ostensibly offer superior benefits over traditional SAP automation efforts whereby SAP transactions are recorded from scratch, they also have potential drawbacks that are often obscure and can hamper automation progress. Many of the drawbacks from SAP testing accelerators are overcome with what is known as a “next-generation accelerator.”

BACKGROUND

Test accelerators refer to prebuilt and generically recorded test cases that could be used to test packaged enterprise business applications. Accelerators typically provided most, if not all, elements necessary to test an entire end-to-end business process such as order-to-cash. The original thinking was that if a single application was deployed at many locations, a single out-of-the box library of prerecorded test cases that can be modified as needed would accelerate the implementation of test automation by providing prebuilt content in a proven framework to the SAP user community.

*Contributed by Arsin Corporation.
Initial test accelerators’ assets focused on screen logic, screen elements, and test scripts. This made it possible for companies to reconfigure and edit these preexisting assets to reflect the unique configuration settings established at each SAP installation. This further allowed test developers to greatly reduce the effort associated with building a test asset development framework and automated test cases from scratch.

This reusability of test assets was a tremendous benefit to the test script developer. By reusing the fundamental screen elements it was possible to quickly put together many different test scripts in a short period of time.

**CHALLENGES**

While test accelerators were an improvement over traditional automation efforts of developing automated test cases from scratch, they still suffered from the following four main problems:

1. Limited system validation
2. Increased maintenance
3. High costs
4. Complex data management

Traditional SAP test accelerators do not embed sufficient programming logic for validating business processes or validating processes at the back end of the application.

An effective test acceleration solution must incorporate test asset maintenance in its thinking. When we say maintenance, we are referring to change management and control. Changes are a natural part of any business process and these changes percolate down to the test assets as well. For a test accelerator to be effective, it must contemplate this reality and provide a solution to easily manage, modify, and evolve with the changing SAP business processes.

The current model most graphical user interface (GUI) test tool providers use for managing data for a test script is a spreadsheet. For each SAP transaction in a test script they will associate a test script to input the data and another spreadsheet for validating the results of that transaction. Exhibit B.1 is a diagram of an order-to-cash (OTC)
EXHIBIT B.1 Decomposition of Order-to-Cash Scenario
end-to-end scenario encompassing multiple SAP transactions strung together. The SAP test accelerator will offer a series of automated test cases for each transaction linked together to form a single test script for the complete business process. Each transaction requires a spreadsheet to drive the execution. It is likely that the end-to-end process in Exhibit B.1 for OTC will have over 20 spreadsheets associated with it. Considering that an organization may have 20 different OTC scenarios that must be tested, it is possible to have hundreds of spreadsheets containing the test data for just OTC.

Existing SAP test accelerators have prebuilt libraries that are generic and therefore any economies of scale are limited. For example, every time a test scriptwriter constructs an automated test case for entering an order through an SAP transaction such as VA01 (for sales order creation), it is largely a unique activity subject to the specific SAP configuration settings under which the process was automated. When one multiplies this effort across all the transactions that are part of a typical SAP end-to-end scenario, it becomes obvious that there is a lot of labor involved in constructing and modifying automated test cases from the SAP test accelerators. Current tool vendors do not want to point this out because they want to sell you their tools and SAP test accelerators. Service vendors do not want to point this out because they would rather maximize their profits from billable hours associated with supporting and maintaining test cases derived from SAP test accelerators.

AN ENHANCED APPROACH

Now that we have identified some of the issues with the current paradigm of SAP test automation and first-generation test accelerators, let us look at how one assembles a better solution through next-generation SAP test accelerators. We will look at new test script creation methods, new methods for managing changes to test assets, new concepts for managing test data, more efficient techniques for performing lights-out testing, and a different cost model that makes test automation generate a respectable return on investment (ROI), all of this done within the context of a new SAP-centric test acceleration paradigm. Furthermore, the new paradigm includes the concept of a
labor cost model that consists of one-time test case automation that is distributed everywhere.

One-time test case automation that is delivered everywhere implies there is an inherent leverage in every test case that is automated. In next-generation test accelerators, this is accomplished through the use of test components. Test components can be thought of as automated test cases that have the functionality to test all of the configuration permutations of the SAP transaction that they test. For example, the same test component for SAP transaction VA01 can be used to test SAP transaction VA01 at various SAP implementations regardless of the SAP configuration settings for transaction VA01. This aforementioned test component could then be reconfigured to mirror the specific configuration of VA01 at each SAP installation. The labor associated with the construction of the test component is then distributed across various SAP implementations that have SAP transaction VA01 as part of their functional scope. Test components are capable of testing all the different configuration settings of an SAP transaction.

Each test component corresponds to a SAP transaction code so a succession of components can be quickly strung together to test the end-to-end business process. To customize this sequence to your specific SAP configuration, the test developer selects from a table the screens used in a given transactions and the fields used for each screen. In other words, they configure the component to match the configuration of the transaction code.

There are many benefits to an automated test library for SAP that is constructed of transaction-level components. The first is naturally the cost as the leverage of automate-once and distribute-everywhere is intuitive. Another, equally important benefit is the implied framework inherent in its structure. The configuration and implementation raises the discussion from a technical, GUI test tool level up to a business-process level more common with the tenets of SAP. A third benefit is that this implied structure allows for changes to test assets in a much more familiar and comfortable manner. Changes such as field additions or deletions and screen additions or deletions do not need to happen at a code level but through a forms-based selection process, thus eliminating test script authoring entirely from the process. This simple change reduces cycle time dramatically by reducing the
mental processing time needed in authoring a test script. In fact, no test script authoring is necessary. The time savings, efficiency, and accuracy for configuring these test components versus constructing an automated test case are analogous to your schoolday preference for taking a true/false test versus an essay test. In short, it is just faster, cheaper, and better.

Next-generation accelerator pricing has driven the cost of SAP test automation down to the cost of approximately one person-year of effort to cover the majority of the core SAP critical business processes. Even for the smallest of installations, the gain of efficiency is hard to summarily dismiss as too expensive. Even the most skeptical are wise to take a closer look.

In a next-generation accelerator test, components not only accommodate test execution but they perform validation as well. By building validation into the test component, a tester does not have to perform endless screen reads to retrieve the validation values for a field. A test component simplifies the process of field validation because it knows the location of the field within the SAP database and reads its value directly from the SAP table. It is also possible to identify additional validation elements in other transactions, which may be useful for validating (technically this is validation not verification) the actual results with the expected results. Retrieval of this data can be specified quickly and easily if required to augment the prebuilt validation associated with each transaction code. Most important, it institutionalizes the validation knowledge of the functional experts in the test component. This fact provides the greatest value both in time savings and domain expertise.

By building validation into each test component, there is a significant time savings when constructing an end-to-end test of a business process. Instead of working at a test script level, the construction is done at a business level by choosing the transaction that needs to be tested without much effort focused on the validation of the data, since it is built into the test components.

This technique is possible only through the use of a SAP test accelerator, which leverages off components to create the test scenarios and a next-generation test accelerator as it permits the use of validation that is built into each component. The means by which these components retrieve data from the SAP database is through the use of a validation engine. A validation engine is a software mechanism
that permits direct access to the SAP database to retrieve values. It works in conjunction with a test component library that makes a data retrieval request of the validation engine. As a test is being executed, a test component starts the execution of a test script inside the GUI test tool. That execution goes to a screen transaction displayed in the SAP GUI and inputs the execution data for the transaction. This causes the step-by-step execution of one or more SAP transactions depending on the complexity of the test script. After each transaction is completed, the test script makes a request of the validation engine, asking for the values necessary to validate the results of the transaction just completed. These values are returned to the GUI test tool, compared with the expected results, and a pass or fail value is assigned to that test step. (See Exhibit B.2.)

A validation engine can also increase the efficiency of testing in- and outbound interfaces to SAP. Interface testing can be done through a GUI test tool but it requires the skills of a test script writer, a Visual Basic programmer, and an ABAP program in order to generate the code to access the internals of SAP through a test tool. Using a validation engine simplifies this process greatly by eliminating the need for an advanced business application programming (ABAP) programmer as well as a Visual Basic programmer, and once a test component is in place it can be reused in other end-to-end test scenarios with little technical expertise. Validation engines greatly simplify the

**EXHIBIT B.2** Validation of Processes through the GUI and Back End with Validation Engine
effort required to build end-to-end test scenarios that cross over multiple platforms.

Through the use of validation engines for next-generation SAP test accelerators, building lights-out test automation is a much simpler proposition and realistically attainable through the framework and structure provided in a next-generation accelerator. By enabling this efficiency, the real ROI begins to appear to test automation naysayers and the benefit of faster cycle times, deeper testing, and lower costs can be realized.

MAINTENANCE AND FEATURES OF ACCELERATORS

As previously mentioned, with the use of next-generation test accelerators it is possible to think of the automated test case for a particular transaction as a test module or test component. That single component is then used by larger automated test cases for end-to-end processes that consist of multiple SAP transactions so that maintenance or changes to that individual component are propagated across all automated test cases that use that component. This simplifies the change process and reduces the labor associated with keeping test assets current. First-generation test accelerators solved the challenge of constant change with the use of this component architecture but in so doing introduced a number of other challenges that were subsequently addressed by next-generation test accelerators.

The challenge with building test scripts from components is the management and coordination of these test components across a group of test developers. Like any development process without a system for tracking, versioning, and distributing these test assets, the overall system effectiveness is greatly inhibited. Without a centralized tool or method for managing this resource, an organization can get into trouble very quickly.

One method for controlling these assets is to treat them as you would any software asset and use a source code control program to manage the distribution and control of these assets. Next-generation test accelerators are implementing central access to these assets through the authoring environment while still using a source code control program for overall management. This provides easy access, use, and reuse to the test developers, at the same time providing all of
the benefits of an asset that is governed by the rules in a source code control program. The source code control program provides a repository for the test component library as well as a copy of the database that contains the configuration data of the metadata, which are the elements and objects that make up the automated test cases. The overall impact is a disciplined test environment with the ability to retrace past tests that assures consistent testing and manages the ongoing evolution of your test assets in a controlled environment.

With the management and control of these test assets defined, let us look at how changes are made. When working with a next-generation test accelerator most process changes are taken in stride, as they no longer require the intensive coding changes. Working with metadata through a forms selection process simplifies the testing process greatly. However, there are times when the change to your SAP system is not based on standard SAP transactions, but is based on a modification that your organization has made to SAP or perhaps interfaces to another application altogether. In this case your test accelerator needs to be able to handle custom Z-transactions or inbound or outbound interfaces. If your test accelerator is designed properly, it is possible to build custom components to address the specific needs of these custom objects. It is very common to run into many custom objects that are part of core, critical business processes, so you should assume you will need to deal with these.

In SAP, Z-transactions exist because the necessary functionality in a given SAP transaction did not exactly meet the needs of the user; therefore it is unrealistic to think that a test accelerator will have the necessary functionality to exactly test that Z-transaction. Similarly, an interface to another system will not be included in any standard test accelerator, so a custom test component will be necessary.

To implement effective test automation in these environments, a test accelerator provider must either provide effective training in developing test components or provide an impeccable service to build them for you, or more likely both. If for no other reason than to have the freedom of choice, a vendor must supply a developer's course on the construction of test components. This course should provide enough detail of the internals of the component so that it can be included in the library of other test components and operate seamlessly within that environment. Additionally, the accelerator vendor should be capable of providing a service to build these components for you,
should you choose to outsource their creation. Eventually, we will see third-party service providers offering test component development.

Next-generation test accelerators that store execution, validation, and metadata in a relational database are a dramatic improvement over managing test data in spreadsheets. By centralizing data the test platform simplifies test creation, manages access control, simplifies backup, recovery and revision control, and enables simplified audit and compliance with government regulatory requirements under Sarbanes-Oxley, the Federal Drug Administration (FDA), and others.

Initial SAP test accelerators have been a promise for a number of years but their value is hampered on account of the following five major challenges:

1. Test script creation slowed the adoption as many were not ready to follow the paradigm of the test tool provider over SAP’s business process model.
2. Most test organizations were in the dark about lights-out testing, so they were unable to see the ROI on test automation, even with accelerators.
3. Maintainability of test scripts was a significant hindrance for anyone who was writing test scripts manually and required an army of staff to keep test assets current.
4. Data management was unsecured and this only exacerbated the maintainability of test assets.
5. Finally, the cost model for building test assets was based on a custom programming model instead of a build-once, distribute-everywhere model that distributes the cost across the entire SAP community.

Times have changed and forward-thinking test automation companies have solved these five key problems, so if one has been skeptical of SAP test accelerators in the past it is a good time to take a closer look at the state of the industry.