Technology

Testing Innovations
Improving Requirement and Test Quality with Model-Based Testing
Quality directors and test managers are continually seeking ways to improve their organization’s testing productivity and time to market, while reducing testing costs. However, there are obstacles to achieving these goals. Many testing executives have a limited view of existing test coverage for an application. Others struggle with the creation of meaningful test cases due to testers’ limited business knowledge. And, in many cases, companies do not have a systematic or reproducible way of deriving test cases from requirements.

One way to avoid these obstacles is model-based testing. Unlike standard application testing, model-based testing tightly links test case creation and maintenance to business processes. The approach is already used in the communications, security and automobile industries to test protocols. Now it is gaining momentum in the application development life cycle.

Defining model-based testing

Model-based testing in its wider definition encompasses all approaches of driving an efficient testing model that maximizes test coverage. It includes a defined and reproducible way of deriving actual test cases from the model. Such a model could be a set of business rules and data dependencies as used as input for statistical test methods. In a stricter definition for discussion in this document, model-based testing is the use of a flow model (visualized in a directed graph1, such as a business process flowchart or state diagram) to define all testable functionality as well as specific verification points, from which actual test cases are derived. This application of model-based testing can verify if the system under test correctly provides this control flow per the verification points. Possible flow models include:

- A control flow model like a business process as defined by a business department.
- An event flow model, which describes events and event interactions of a graphical user interface.
- A state diagram, which represents a behavioral model composed of a finite number of states associated to transitions or sets of actions. State diagrams are often used in the telecommunications and automotive industries for testing communications protocols.

Model-based testing can be used in a stand-alone testing service, such as when the application implementation is not part of the work, as well as for systems integration projects in which testing is one component. The approach is also useful as a starting point for requirements collection, including documenting legacy systems for which other documentation is not available. In this event, documentation and test preparation can be combined into one step: creation of the model. The resulting model can then serve as both the business process documentation and input for test case generation.

However, model-based testing cannot be used in every situation. To decide if the approach will work for your company, see the sidebar entitled “Determining if Model-based Testing Is Right for Your Organization.”

---

1 For more information about a directed graph, see “Components of a Graphical Model” sidebar.
Model-based testing depends on directed graphs as input. Modeling techniques vary, but in essence it comes down to: “vertices connected by directed edges.”

A vertex, for example, represents an activity in a business process, or a state in a state diagram.

An edge connects two vertices and has a direction (“arrow”), so that a flow is established, which defines which activity or state can potentially follow the current activity or state. In a state diagram an edge represents the activity, which is required to move the system from one state to the other.

Essentially, model-based testing tools begin at a defined starting point and walk along the edges, a procedure called traversing. Depending on the model and tool technique, vertices and/or edges become test steps in the resulting script.

Components of a Graphical Model

Three Perspectives on an Innovative Idea

Model-based testing can benefit various roles in the application development lifecycle as follows:

- Business analyst—Model-based testing helps business analysts review test cases more accurately to ensure they are complete. Instead of reviewing an abstract list of test cases, business analysts can look at graphical models in easy-to-understand diagrams that showcase the business process.

- Test designers—Since model-based testing generates test cases, test designers can move away from writing down individual test cases to the more creative task of modeling based on actual business scenarios.

- Quality assurance director—Unlike traditional or risk-based testing approaches, which cannot cover all possible test cases, model-based testing allows quality executives to view the theoretical maximum of possible tests. This equips them with better knowledge about how to approach and control test coverage, and makes selecting and assigning tests much easier.
Understanding the different approaches

How does traditional testing differ from model-based testing? In traditional testing, test designers document only the test cases—the outcome of their individual understanding of how an application should function. (See Figure 1.) The result is a text-based list of test cases, which is more difficult to verify for completeness than it is to check a graph. Also, the text-based list does not document how these, and only these, test cases were derived. Their link to requirements is set manually, which means that a test designer must verify during a review that a set of test cases actually covers a particular requirement.

In contrast, with model-based testing, test designers document their understanding of the business processes by starting with a business scenario and creating a graphical representation of the test model to represent how the system should behave. (See Figure 2.) A tool can then generate the actual test cases, taking the graph as input. As such, the thought process behind how to develop test cases becomes an artifact that can be stored for future use.

Figure 1: Traditional testing

The login screen asks for a username and password. Successful login requires that the password matches the one stored in the database for the account identified by the given username.

Test Designer

Test Cases
1. Success
2. Wrong User
3. Wrong Password

Figure 2: Model-based testing

The login screen asks for a username and password. Successful login requires that the password matches the one stored in the database for the account identified by the given username.

Test Designer

Test Case Generator

Test Cases
1. Success
2. Wrong User
3. Wrong Password
Primary advantages of model-based testing

In most cases, the graph of the model looks like a business process, making it easier for test designers to discuss and review with business analysts the variations or paths that exist within the overall process flow. These intense reviews give the test designer the opportunity to fine tune the model to accurately reflect the requirement. Similarly, peer reviews of the test model become more efficient.

A major advantage of these reviews is that they can reveal missing details in the requirements, especially in the areas of branch and validation concerns. Since this step is conducted at an early stage, it can significantly increase the quality of requirements, in some cases leading to defects being raised around ambiguous or inaccurate requirements. Thus model-based testing is an important component of a high-maturity testing approach and a holistic, end-to-end quality strategy.

After the review is complete, the test designer has a final, precise model from which to derive test cases. The testing team can then feed the model in form of a graph into a test case generator tool, which traverses the graph based on given parameters, to create a comprehensive list of test cases and scripts. Traversal refers to the process of visiting or examining each vertex and/or edge in a graph in a systematical way. (See Figures 3 and 4.)

Figure 3: Model-based testing input diagram, here in the form of a state diagram
An input similar to the graph in Figure 4, which represents possible states and activities on a login form, can result in the following test script, which has been created using generator setting “complete edge coverage”\(^2\) to make sure all edges (i.e., activities) are used at least once:

- Start
- Login field “User”
- Enter invalid User
- Login field “Password”
- Enter invalid Password
- Submit Login
- Password invalid
- Message: Login Error
- Dismiss Login Error Message

- Login field “User”
- Enter valid User
- Login field “Password”
- Enter valid Password
- Submit Login
- User and Password valid
- Message: Logged in
- Logout

The coverage as generated is depicted in the diagram in Figure 4. Please note that the script can be executed in one step from the beginning to the end. The three colors can be interpreted as three distinct test cases, but are mostly meant as reading support.

---

\(^2\) Complete edge coverage is the most often used of many possible test case generation strategies. The generator, which creates the test script, makes sure that every edge of the given graph is used at least once, in other words it is covered. Good generators try to find the minimum number of test cases required to cover each edge once.
Model-based testing is both an innovative testing approach and a quality activity that begins upstream of the testing process. A comprehensive review cycle between test designers and business analysts is built into the approach, which can reveal missing details and significantly increase the quality of requirements in an early stage.
Additional advantages

In addition to finding flaws early, model-based testing offers other advantages to further reduce the cost of poor quality. Although the details are specific to the selected tool, testing teams can typically expect the ability to do the following:

Structure complex graphs into subgraphs (“zooming”) and merge these during generation time.

Attach priorities or weights to edges and/or vertices of the test model, enabling quality assurance directors to select specific test cases and more accurately control intended test coverage.

Track requirements, at least using references within the test model, but depending on the tool’s features also by integration with requirements management software.

Visualize test coverage. In tools, which are well integrated with a graph editor, it is often possible to highlight the paths, which are covered by the generated test cases such as shown in Figure 4.

Integrate with test management tools for storing generated test scripts, while tracking script execution and test results is then left to the test management suite.

Use different generation strategies. Once test models become larger it is no longer efficient or necessary to calculate every possible test case, or path through the model. Hence, model-based testing generators allow different strategies to calculate the minimal number of test cases or test steps to satisfy a given condition.

Support test execution automation:

• Model-based testing tools support the use of templates, which describe the way the generated output is formatted, so that the same generated test script can be stored as a spreadsheet (for later manual execution) or as executable code in any preferred test automation language.

• Each output element becomes a keyword, which will need to be implemented by a test automation engineer as usual. Once the library of automated keywords is available even the test execution of the generated scripts can be automated.

• Note that with detailed test models, such as the example above, the automation effort is reduced as the individual keywords are simple and can be reused quite often in different graphs and activities (“Enter”, “Submit”). However, if the test model has not yet reached such a level of detail it still can be a basis for relevant test cases, especially for regression testing of a complex system.

Conduct online vs. offline testing. Most tools allow generating a script and outputting it in an offline format for later execution (manually or automated) such as in the example above. Some tools also allow an online testing mode, in which the test model is interpreted at runtime, actions are executed against the system under test and edges of the model are chosen based on the system’s actual response. This is especially useful for long-running tests like performance, volume or memory usage (garbage collection) tests.

Manage requirements changes. The graphical representation of the test model will be easier to modify than a long list of test cases, allowing the testing team to speed up the re-creation of test cases if specific aspects of the application need to be retested due to changes.

Challenges of model-based testing

Model-based testing works quite differently from traditional ways of creating test cases manually. Hence, organizations may face the following challenges:

Education of test designers—Test designers will need to learn how new tools like graph editors and generators work. They will also need to embrace thinking in models and graphs rather than in distinct test steps. This is a level of abstraction that will require additional training for some testers, though in general graphical representations are easier to understand than text. For an initial exploration into model-based testing, select motivated people who like to learn new techniques and ideally have a basic programming or modeling background.

Selection and integration of tool suite—While the theory of model-based testing works on paper, it is only efficient if backed by tools. Various commercial and open source tools are available in the market. Some of them are specialized to certain types of systems or testing standards; others follow a more generic approach. The tool selection will depend on the system under test, the organization’s existing tool landscape, and the preferences and technology-savvy of test architects and designers.

Maintenance of flow model—Creating the first graph of a test model is often seen as a valuable and productive task. However, the larger the graphs become, the more difficult it will be to maintain them and keep them consistent. Especially for larger systems it is advisable to define from the beginning a clear strategy of layers in the model. Wherever possible keep the organization and layers of the test model aligned with how the business requirements or documentation is structured.

Integration with existing test approaches—In non-green field developments, model-based testing will in the beginning probably only replace parts of an organization’s tests. Testing executives will need to define which aspects to replace and how the test model integrates with other test approaches.
Benefits of model-based testing

Organizations can overcome these challenges with a strategic approach to model-based testing, which can raise test quality in several ways and also provide excellent support for test maintenance as requirement changes are easily modeled. This minimizes the impact on generated test conditions. Specific benefits of model-based testing include:

- **Increased quality of requirements in early stage**—In the traditional application development life cycle, requirements are formalized in a late stage when programmers create code according to the requirements. The formalization happens because coding does not allow for vague statements and requires specific answers about how to proceed if a condition is not met.

  Conversely, model-based testing helps to formalize and improve the quality of requirements much earlier in the life cycle when test designers review graphical representations of models with business analysts. To create a model, test designers must have clarity about conditions and determine exact parameters in a formalized diagram. Instead of translating vague requirements, test designers can discuss the requirements with business analysts. Incorporating this feedback loop into the process allows test designers to clarify and have the requirements changed, while also capturing deeper levels of detail in the model.

- **Automated test case creation**—In standard application testing, the testing team undertakes the time- and resource-intensive task of creating reproducible and detailed test cases. However, with model-based testing, the process is automated and test cases are generated through a software tool. With this approach, test designers can take the time to create and vet the right model for each business scenario, which will cover a wider range of test objects in the same amount of time.

- **Improved test coverage management**—Since it is difficult to test all possibilities, quality directors and project leads must rely on test coverage metrics to indicate how much of and how well a system is tested. Model-based testing allows for test coverage to become pre-defined and controllable through parameters. Since the model covers all requirements, organizations can create automatic test conditions to cover an entire application, and then prioritize and choose which test cases to run. In this sense, quality assurance directors can fully assess test coverage with the model and achieve the highest possible testing percentage in the time allotted for testing.

- **Increased efficiency in testing effort**—Since the model-based approach to testing can significantly improve defect detection efficiency over standard testing approaches, testing teams can save time in the actual testing effort.

- **Expanded testing career path**—With model-based testing, the test designer’s role is enhanced from a test case producer to a test modeler. Organizations can use this approach to broaden career path options for testing professionals, elevating the role to the equivalent of a business analyst. In some cases, organizations may want to split the role into two paths—one for test modelers, who wish to expand their skill set, and a second for test executors, who are content with their career position or are just starting their career.
Moving ahead with model-based testing

Testing executives who are curious about model-based testing—or ready to proceed—have a few options available to achieve these benefits. To move ahead internally, organizations must select and install a new model-based testing tool suite, and train a core group of testing designers in the requisite skills for model-based testing. Since the approach is different from the standard user interface testing, testing teams need to think about training and skill development that teaches test designers how to document their thinking process into usable graphical models.

Another way to try model-based testing is to work with a strategic testing partner, and outsource the testing component to a group of trained testing professionals with model-based testing experience. To mitigate risk, seek a provider who focuses on quality and uses industrialized testing methods to achieve efficiencies across the end-to-end testing life cycle.

In either case, it is usually best to start with a proof-of-concept project or to pilot model-based testing on a low-risk application. (See Figure 5.) Review the quality and testing value generated through model-based testing, and then proceed toward using the approach in test execution.

Figure 5: Levels of maturity of model-based testing

Create test model → Introduce test case generator → Integrate with test management tools → Trace requirements → Generate automated scripts

Minimum for model-based testing

Put model-based testing to the test

As an innovative approach, model-based testing can raise the quality of the application your testing professionals are reviewing—from the requirements phase, all the way through testing, acceptance and production—while lowering costs, improving productivity and reducing time to market.
Model-based Testing Yields 25 Percent Improvement in Testing Effort

An online gaming company sought a better mechanism for testing new releases of its popular games. The company asked Accenture to use a model-based testing approach for test automation. Accenture testing professionals converted the client’s use cases and requirements into easy-to-understand graphical models and then generated test cases using model-based testing tools. The automation framework traversed the flow of testing execution across 225 different paths as opposed to only one flow in a traditional testing approach. In addition, the team’s model-based approach significantly improved defect removal efficiency over standard automation and saved 25 days in the testing effort. (See Figure 6.)

Today, the online gaming company is continuing to use model-based testing to maximize test coverage for its games.

Figure 6: Case study of time savings achieved with model-based testing

<table>
<thead>
<tr>
<th>Baseline Effort</th>
<th>Actual Effort</th>
<th>Net Saving Delivered</th>
<th>% of effort saved</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>75</td>
<td>25</td>
<td>25%</td>
<td>Completed 25 days earlier using Model Based Testing approach</td>
</tr>
</tbody>
</table>
## Determining if Model-based Testing Is Right for Your Organization

Should your organization incorporate model-based testing into its testing approach? To answer this question, review the self-assessment checklist below. Accenture can help assess your unique business situation and review the system under test as well as the specific requirements to determine if model-based testing can be used to increase test coverage and improve testing productivity. Accenture can also help your organization select the appropriate testing tool vendor and organize system test in a model-based way.

### Application development life cycle maturity
- **Maintain Current Testing Approach**: Immature (TMMi level 1)
- **Pilot Model-based Testing**: Mature (TMMi level 2 or higher)

### Quality management/requirements management testing
- **Maintain Current Testing Approach**: Development team conducts testing
- **Pilot Model-based Testing**: Dedicated, centralized testing team for key applications

### Cost of quality
- **Maintain Current Testing Approach**: Cost of poor quality is not measured or is not fully known
- **Pilot Model-based Testing**: Methodical improvement plans in place to improve quality

### Client vision
- **Maintain Current Testing Approach**: Unclear vision on future testing approach or plans to continually improve testing process
- **Pilot Model-based Testing**: Pursuing measurable plan to industrialize testing, perhaps in a testing center of excellence model

### Ability to absorb change
- **Maintain Current Testing Approach**: No established change management or training resources
- **Pilot Model-based Testing**: Executive sponsorship; change management and training resources available

### Process/data/business logic complexity
- **Maintain Current Testing Approach**: Highly complex application and processes
- **Pilot Model-based Testing**: Simpler application and process make it easier to create models and pilot model-based testing

### Release management maturity
- **Maintain Current Testing Approach**: Little to no versioning control
- **Pilot Model-based Testing**: Mature versioning control methods that can be applied to test models

### Answer these questions for a specific application where model-based testing could be piloted

<table>
<thead>
<tr>
<th>Question</th>
<th>Maintain Current Testing Approach</th>
<th>Pilot Model-based Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application development life cycle maturity—How mature are your system development processes? How strictly are processes enforced? Is change handled in a formal way?</td>
<td>Immature (TMMi level 1)</td>
<td>Mature (TMMi level 2 or higher)</td>
</tr>
<tr>
<td>Quality management/requirements management testing—Do you conduct testing internally as part of development or do you have a dedicated testing team?</td>
<td>Development team conducts testing</td>
<td>Dedicated, centralized testing team for key applications</td>
</tr>
<tr>
<td>Cost of quality—How much does your organization spend on quality measures in the testing process? What is the cost of poor quality in terms of development or testing rework?</td>
<td>Cost of poor quality is not measured or is not fully known</td>
<td>Methodical improvement plans in place to improve quality</td>
</tr>
<tr>
<td>Client vision—How solid is your organization’s testing vision?</td>
<td>Unclear vision on future testing approach or plans to continually improve testing process</td>
<td>Pursuing measurable plan to industrialize testing, perhaps in a testing center of excellence model</td>
</tr>
<tr>
<td>Ability to absorb change—How quickly can your organization move toward model-based testing with activities such as managing change throughout the testing community and updating testing resources training?</td>
<td>No established change management or training resources</td>
<td>Executive sponsorship; change management and training resources available</td>
</tr>
<tr>
<td>Process/data/business logic complexity—How complex are your organization’s processes, data and business logic for the system under test?</td>
<td>Highly complex application and processes</td>
<td>Simpler application and process make it easier to create models and pilot model-based testing</td>
</tr>
<tr>
<td>Release management maturity—What controls does your organization follow to manage versioning and merging of documentation and software? Are you prepared to handle graphical models versus textual models?</td>
<td>Little to no versioning control</td>
<td>Mature versioning control methods that can be applied to test models</td>
</tr>
</tbody>
</table>
Contacts

To learn more about how Accenture Application Testing Services can help your organization reduce testing costs and improve productivity while embedding quality into the application development life cycle, visit us at www.accenture.com/testing or contact:

Jeff Wilkinson
jeff.s.wilkinson@accenture.com or

John McEvoy
john.c.mcevoy@accenture.com

About Accenture

Accenture is a global management consulting, technology services and outsourcing company, with more than 244,000 people serving clients in more than 120 countries. Combining unparalleled experience, comprehensive capabilities across all industries and business functions, and extensive research on the world’s most successful companies, Accenture collaborates with clients to help them become high-performance businesses and governments. The company generated net revenues of US$25.5 billion for the fiscal year ended Aug. 31, 2011. Its home page is www.accenture.com.