The Test Pattern

Date: 2016-04-15
The Test Pattern

Contents

1 Introduction ........................................................................................................................................ 2
  1.1 Pattern Aims .................................................................................................................................. 2
2 Concepts ........................................................................................................................................... 3
3 Viewpoints ....................................................................................................................................... 4
  3.1 Overview ....................................................................................................................................... 4
  3.2 Rules .............................................................................................................................................. 5
  3.3 Testing Context Viewpoint (TCVp) ................................................................................................. 7
    3.3.1 Description .............................................................................................................................. 8
    3.3.2 Example Testing Context View ................................................................................................ 9
  3.4 Test Set-up Viewpoint (TSVp) ...................................................................................................... 11
    3.4.1 Description .............................................................................................................................. 12
    3.4.2 Example Test Structure Viewpoint .......................................................................................... 13
    3.4.3 Example Test Schedule Behaviour Viewpoint ........................................................................ 14
    3.4.4 Example Test Set Behaviour Viewpoint ................................................................................ 15
  3.5 Test Case Viewpoint (TCVp) ......................................................................................................... 16
    3.5.1 Description .............................................................................................................................. 17
    3.5.2 Example Test Configuration View ........................................................................................... 18
    3.5.3 Example Test Behaviour Viewpoint ......................................................................................... 19
    3.5.4 Example Test Record Viewpoint ............................................................................................ 21
4 Summary ........................................................................................................................................... 23
5 Related Patterns ............................................................................................................................... 23
6 References & Further Reading ........................................................................................................... 23
1 Introduction

Testing is an essential part of any system development. Testing must demonstrate two main aims of the development by answering two questions: [Boehm]

- Have we built the right system? (validation)
- Have we built the system right? (verification)

Both verification and validation (V&V) can take on many different forms so it is important that the system can be tested in many different ways, ideally using a single model as a basis for all the testing activities. In the context of safety-critical systems development it is vital to justify that adequate testing activities have been performed, and sufficient test cases have been exercised on the system under test.

As a consequence it is desirable to adopt an organised approach to define test schedules that define the creation and execution of test cases.

Such an approach must be produced in a systematic way. If test schedules can exploit tool support automating some of the activities involved, then it is essential that these tools conform to such an organised approach, because otherwise unnecessary effort will have to be spent on mapping the artefacts produced by the tools to the evidence required to justify adequateness and comprehensiveness of the test schedule. This model-based approach forms an essential basis for model-based testing.

1.1 Pattern Aims

The Test Pattern defines a structure of the model that can then be used as a basis for model-based testing. The main aims of the Test Pattern are shown in Figure 1.
The Test Pattern

- ‘Define test set up’, that allows the Test Schedule and Test Sets to be defined.
- ‘Define test cases’, that allows the actual tests that are to be executed to be defined.

Notice that the Test Context is so important that it constrains the other two use cases on this diagram.

2 Concepts

The main concepts covered by the Testing Pattern are shown in the Ontology Definition View (ODV) in Figure 2.

![Figure 2 - Ontology Definition View showing Testing concepts](image)

The diagram in Figure 2 shows the Testing Pattern ontology by identifying the main concepts and the relationships between them.

A fundamental part of the ‘Test Case’ Pattern is, quite naturally, the ‘Test Case’. A ‘Test Case’ is executed against a ‘Testable Element’ using ‘Test Data’. One or more ‘Test Case’ is collected into a ‘Test Set’, one or more of which make up a ‘Test Schedule’. It may be possible than more than one Test Set may be required depending on the Testing Context. For example, one Test Set may be required for validation, one for verification, etc. However, it is not good enough to simply define a structure for a ‘Test Case’ and then realise it, as we need to understand why the testing is necessary in the first place, what are the constraints, what other systems are required, what is the level of...
The Test Pattern

testing required and so on. This overall rationale behind the tests is represented by the ‘Testing Context’. The ‘Testing Context’ comprises three main elements:

- One or more ‘Testing Need’. This represents the needs of the testing that will be carried out (for example: verification, validation, etc.) and will also define any constraints on the testing, such as the scope of the tests, any specific techniques, and so on. It should be noted that the ‘Testing Need’ does not represent the needs of the system or any of the other stakeholders.
- One or more ‘Required System’. This represents any system that falls outside the testing boundary that may be required for the successful execution of the tests. Examples of such required systems include: the environment, standards, test data, etc.
- ‘Testing Boundary’. The ‘Testing Boundary’ represents the partition between what is included on the model (that satisfies the ‘Testing Need’) to be used for the testing and everything that is not part of the model, but is necessary to perform the testing (the ‘Required System’).

The ‘Test Case’ itself tests one or more ‘Testable Element’. This can be any element from the model of the ‘System Under Test’, any collection of elements or, indeed, the whole model. The main aim here is to simply identify the area of the model that will be tested.

The ‘Test Case’ forms the heart of the Pattern and comprises a few key concepts that are important to understand:

- ‘Test Description’, which is quite straightforward and simply defines a text description and a unique identifier for the ‘Test Case’.
- ‘Test Data’, that represents any data that is used as an input to the testing activities.
- ‘Test Configuration’ defines the configuration of one or more ‘Testable Element’ that is to be tested. This is a structural representation of the ‘Testable Element’ and any other model elements that are connected to it in a specific configuration that is required in order to satisfy a ‘Testing Need’.
- ‘Test Behaviour’ that describes an anticipated behaviour of the ‘Test Case’ that is applied to a ‘Test Configuration’. There will usually be more than one ‘Test Behaviour’ associated with each ‘Test Configuration’ and the anticipated behaviour may represent both desirable and undesirable examples.
- ‘Test Record’. Whereas the ‘Test Behaviour’ describes the test that will be performed by defining the expected behaviour, the ‘Test Record’ captures the actual behaviour that is observed during the execution of the Test Case.

These concepts are visualised through a number of Views that are together in the Testing Pattern.

3 Viewpoints

This section describes the Viewpoints that make up the Testing Pattern. It begins with an overview of the Viewpoints, defines Rules that apply to the Pattern and then defines each Viewpoint.

3.1 Overview

The Testing Pattern defines a number of Viewpoints as shown in the Viewpoint Relationship View (VRV) in Figure 3.
The Test Pattern

The diagram in Figure 3 shows that the Testing Pattern has three main Viewpoints:

- The ‘Testing Context Viewpoint’ that defines the rationale for the Viewpoint and identifies a number of needs and constraints that must be satisfied in order that the test campaign be deemed successful.
- The ‘Test Set-up Viewpoint’ describes the structure of the Test Schedule and Test Set (using the ‘Test Structure Viewpoint’) and then the execution of the Test Schedule (using the ‘Test Schedule Behaviour Viewpoint’) and the Test Set (using the ‘Test Set Behaviour Viewpoint’)
- The ‘Test Case Viewpoint’ that identifies the scope of the Test Case (using the ‘Test Configuration Viewpoint’ that defines how the Test Case is set up), the anticipated behaviour of the Test Case (using the ‘Test Behaviour Viewpoint’ that shows the preferred behaviour of the Test Case) and captures the results of the Test Case (using the ‘Test Record Viewpoint’)

These Viewpoints are defined in the following sections.

3.2 Rules

Sixteen rules apply to the Testing Pattern Viewpoints, as shown in the Rules Definition View (RDV) in Figure 4.
The Test Pattern

The Testing Pattern has 16 Rules defined that are shown in Figure 4 and that are described as:

- ‘TP1: A Testing Context must exist that defines every Testing Need’. This is fundamental to the Testing Pattern as it defines the testing Needs for the whole Test Schedule. Without this Rule, none of the others can be successful.
- ‘TP2: Each Testing Need must be satisfied by one or more of the Test Set View or Test Case’. This represents a coverage Rule in that all of the Testing Needs must be covered by the overall Test Schedule.
- ‘TP3: Every external System that is necessary for the testing activities is a Required System’. This ensures that all external systems necessary for successful testing have been identified.
- ‘TP4: Every Required System must be related to at least one Testing Need’. Any Required System that is not connected to a Testing Need is redundant and, therefore, should not feature on this View.
- ‘TP5: Every Testing Need must be related to at least one Required System’. Every Testing Need must be connected to a Required System as the Required System identifies test models, external data sources, test equipment, etc. This connection may be either direct or indirect.
- ‘TP6: There must be at least one Test Structure Viewpoint, Test Behaviour Viewpoint and Test Set Behaviour Viewpoint’. A View for each of the Test Set-up Viewpoints must exist, otherwise the testing activities are not defined.
The Test Pattern

- ‘TP7: Each Test Structure Viewpoint must include a Test Schedule that is made up of at least one Test Set’. There must be at least one Test Schedule as it is the Test Schedules that ultimately own all of the Test Set and hence Test Cases.
- ‘TP8: One or more Test Schedule Behaviour Viewpoint must satisfy the Testing Context’. The overall Test Schedule Behaviours must satisfy the complete set of Testing Needs in the Testing Context.
- ‘TP9: One or more Test Set Behaviour Viewpoint must satisfy the Testing Context as a whole’. The overall Test Set Behaviours must satisfy the Testing Needs in the Testing Context. The Rule is related to the previous Rule, TP8.
- ‘TP10: Each Test Set that appears on a Test Structure Viewpoint must appear as a life line on a Test Behaviour Viewpoint’. This Rule enforces consistency between the Test Structure Views and the Test Behaviour Views by ensuring that the Test Behaviours that are executed are based on Test Sets.
- ‘TP11: Each Test Set that appears on a Test Structure Viewpoint must have at least one Test Case’. Test Sets are not permitted to be empty – they must contain at least one Test Case.
- ‘TP12: Every Test Case that appears on a Test Structure Viewpoint must appear on at least one Test Behaviour Viewpoint’. All Test Cases must be executed at some point during the testing activity otherwise it is redundant and should not appear.
- ‘TP13: Each Test Set must satisfy at least one Testing Need from the Testing Context’. This is a lower-level coverage check that ensures that all Test Sets ultimately relate back to a Testing Need.
- ‘TP14: Each Test Case Viewpoint must satisfy one or more Testing Need from the Testing Context Viewpoint’. This is a lower-level coverage check that ensures that all Test Set Case satisfies at least one Testing Need.
- ‘TP15: Each Test Case that appears on a Test Structure Viewpoint must have at least one Test Behaviour Viewpoint’. All Test Cases must have their behaviour defined.
- ‘TP16: At least one Test Behaviour Viewpoint must exist for each Test Configuration Viewpoint’. All Test Configurations must form the basis of some testing via the Test Behaviour Views otherwise they are redundant.

Note that the 16 Rules shown in Figure 4 are the minimum that are needed. Others could be added if required.

3.3 Testing Context Viewpoint (TCVp)

The aims of the Testing Context Viewpoint are shown in the Viewpoint Context View in Figure 5.
The main aim of this Viewpoint is to provide the context of the testing activity or, to be more specific, to define:

- The ‘Testing Boundary’ of the testing activity, visualised on a use case diagram using a system boundary.
- The ‘Testing Need’ of the testing activity, visualised on a use case diagram using use cases.
- Any ‘Required System’ outside the boundary that is necessary to complete the testing activity. This may include a number of elements, such as the ‘Requirements Model’ that may be necessary for validation, one or more ‘Standard’ that may be required for quality-based testing, specific test data sets, any testing tools needed, etc.

This Viewpoint really drives the whole of the testing activity and will have Rules defined that can be applied to ensure consistency within the Testing Framework.

This Viewpoint can also be used to create a library of testing Use Cases and then to show how these Use Cases may be satisfied. This will be particularly useful for specific testing types, for example, equivalence class partition testing may be realised using a particular set of Views which may then be used as a basis for generating automated tests.

### 3.3.1 Description

The Viewpoint Definition View (VDV) in Figure 6 shows the Ontology Elements that appear on a Testing Context Viewpoint.
Figure 6 - Viewpoint Definition View showing the Ontology Elements that appear on the Testing Context Viewpoint (TCVp)

The diagram in Figure 6 shows the Ontology for the Testing Context Viewpoint.

The main two Ontology Elements in this Viewpoint are:

- ‘Testing Need’, that represents the fundamental need for the whole testing activity.
- ‘Required System’, that represents elements external to the System that are necessary for testing. Examples of Required Systems include: standards, test equipment, test data, etc.

The Required System and the Testing Need are related via the Testing Boundary.

3.3.2 Example Testing Context View

An example of the Testing Context View is shown in Figure 7.
Figure 7 - Example Testing Context View

The ‘Testing Context View’ in this example is visualised in SysML using a use case diagram.

The main aim of this View is to provide the context of the testing activity or, to be more specific, to define:

- The ‘Testing Boundary’ of the testing activity, visualised on a use case diagram using a system boundary
- The ‘Testing Need’ of the testing activity, visualised on a use case diagram using use cases
- Any ‘Required System’ outside the boundary that is necessary to complete the testing activity. This may include a number of elements, such as the ‘Requirements Model’ that may be necessary for validation, one or more ‘Standard’ that may be required for quality-based testing, specific test data sets, any testing tools needed, etc. Each Required System is represent by an actor on the use case diagram.

On top of these elements there will also be a number of relationships between the different elements, such as:

- Relationships between the use cases, typically the standard SysML dependency-based relations (include, extend, constrain, etc.)
- Relationships between the use cases and the external actors, using the standard SysML associations.

In this example, the main aim of the Testing Context is to ‘Test System’ which has two inclusions:

- ‘Validate system’ which in this case is further specified by stating that in order to validate the system, there are two aims, which re to ‘Satisfy all use cases’ and to ‘Satisfy all contexts’.
- ‘Verify system’ which in this example has two specific types of testing which are ‘Perform functional testing’ and to ‘Perform QA testing’. This aim is further qualified by stating that the verification should ‘Apply at all levels’ which are to apply: ‘...at unit level’, ‘...at integration level’, ‘...at system level’ and ‘...at enterprise level’.
This View really drives the whole of the testing activity and will have rules defined that can be applied to ensure consistency within the Testing Pattern.

The diagram in Figure 8 shows how one of the use cases from the Testing Context View has been expanded to show more detail. In this example, the ‘Satisfy all use cases’ use case has been broken down into three lower-level use cases that describe more-specific testing criteria associated with the use cases.

### 3.4 Test Set-up Viewpoint (TSVp)

The aims of the Test Set-up Viewpoint are shown in the Viewpoint Context View in Figure 9.
The main aims of the ‘Test Set-up Viewpoint’ are to:

- Identify one or more ‘Test Set’ and ‘Test Case’ that make up the ‘Test Schedule’.
- To show how the ‘Test Schedule’ is performed by defining the order of execution of one or more ‘Test Set’.
- To show how each ‘Test Set’ is performed by defining the order of execution of one or more ‘Test Case’.

The Test Set-up Views do not describe the tests themselves but rather the execution of the Test Schedule and its associated Test Sets and Test Cases. The totality of the test set-up Views forms the basis of the test campaign.

### 3.4.1 Description

The Viewpoint Definition View (VDV) in Figure 10 shows the Ontology Elements that appear on a Test Set-up Viewpoint.
In Figure 10 it can be seen that this View has three types – the ‘Test Structure Viewpoint’, the ‘Test Schedule Behaviour Viewpoint’ and the ‘Test Set Behaviour Viewpoint’.

- The ‘Test Structure Viewpoint’ represents the whole of the testing in its entirety and defines the ‘Test Schedule’. Each ‘Test Schedule’ that is identified is further described by defining its component set of one or more ‘Test Schedule’.
- The ‘Test Schedule Behaviour Viewpoint’ represents the highest level View that describes the order that various ‘Testing Set’ are executed in and the conditions under which they are executed.
- The ‘Test Case’ specifies the behaviour for a single Test Case.

The combination of these three Viewpoints makes up the core testing activities.

### 3.4.2 Example Test Structure Viewpoint

An example View that conforms to the Test Structure Viewpoint is shown in Figure 11.
The ‘Test Structure View’ will be visualised in SysML using a block definition diagram showing:

- The ‘Test Schedule’, represented as a SysML block stereotyped by «Test Schedule», and that is made up of a number of ‘Test Set’.
- One or more ‘Test Set’, represented as SysML blocks that are stereotyped by «Test Set» and that are aggregated into the ‘Test Schedule’.
- One or more ‘Test Case’, represented as SysML operations that are stereotyped by «Test Case».

Dependencies between Test Sets can also be used to show where one Test Set may depend on the successful execution of another Test Set.

The example here shows that the ‘Overall Test Schedule’ is made up of two Test Sets, which are:

- ‘Validation Test Set’ which has two Test Cases that are represented by the operations on the block and that are: ‘context coverage’ and ‘use case coverage’.
- ‘Verification Test Set’ which has two Test Cases that are represented by the operations on the block and that are: ‘functional test’ and ‘QA test’.

The dependency shown here represents the fact that the ‘Verification Test Set’ cannot be carried out without the ‘Validation Test Set’ having been carried out.

3.4.3 Example Test Schedule Behaviour Viewpoint

An example View that conforms to the Test Schedule Viewpoint is shown in Figure 12.
The diagram in Figure 12 shows an example of a Test Schedule Behaviour View. The diagram shows the order of execution of a number of Test Sets.

The ‘Test Schedule Behaviour Viewpoint’ may be represented in SysML as a sequence diagram where each ‘Test Set’ is visualised using a life line, and the ordering between them is shown using messages. In order to show the different paths of execution, then combined fragments may be necessary.

As each ‘Test Schedule Behaviour Viewpoint’ defines a scenario, it is possible to define a number of these Views (for example showing what happens when all test sets are passed, or when some fail).

3.4.4 Example Test Set Behaviour Viewpoint
An example View that conforms to the Test Behaviour Viewpoint is shown in Figure 13.
The diagram in Figure 13 shows an example of a Test Set Behaviour View. This View shows what happens inside each Test Set when it is executed in terms of the order that Test Cases are executed and any conditions that may apply.

The behaviour of each Test Set is represented in SysML using a state machine diagram and each Test Case is shown as an activity within a state.

The example here shows the order that the Test Cases must be executed in. This case the ‘functional test’ Test Case is executed first and only upon its successful completion is the ‘QA test’ executed. In the event that the test fails, then the testing is halted and does not continue. This also explains the dependency in Figure 11.

3.5 Test Case Viewpoint (TCVp)

The aims of the Test Case Viewpoint are shown in the Viewpoint Context View in Figure 14.
The Test Pattern

Figure 14 - Viewpoint Context View showing Test Case Viewpoint aims

The diagram in Figure 14 shows the Context for the Test Case Viewpoint.

The main aims of this Viewpoint are to ‘Define test cases’, which includes:

- ‘Define Test Configuration’ that allows the Test Cases to be set up and that includes identifying the Testable Elements that will be the subject of the tests (‘Identify testable elements’ and the relationships between them (‘Identify connections’).
- ‘Define test behaviour’, where the behaviour for each Test Case is defined and that must exist in order for the Test Results to be captured.
- ‘Record test results’, where the information captured during the execution of the Test Cases is recorded.

The ‘Test Schedule’ is shown as an actor on this diagram.

3.5.1 Description

The Viewpoint Definition View (VDV) in Figure 15 shows the Ontology Elements that appear on a Test Case Viewpoint.
The diagram in Figure 15 shows the ‘Test Configuration Viewpoint’ that identifies the part of the model that is under test - the ‘Element Under Test’.

The ‘Test Behaviour Viewpoint’ defines the behaviours for the ‘Test Configuration’.

The ‘Test Record View’ will not be considered in great detail here, as it uses the same visualisation as the ‘Test Behaviour View’.

### 3.5.2 Example Test Configuration View

This section shows two examples of how a Test Configuration View may be visualised. Each of these Views may be visualised using a SysML block diagram.
The Test Pattern

Figure 16 Example Test Configuration View

The diagram in Figure 16 shows an example of a Test Configuration View based on the Stakeholder Roles that have been identified as relevant to the testing activities. Each Stakeholder Role is represented by a SysML block. These will form the basis for identifying any Contexts that are required for the Test Case Behaviour Views.

Figure 17 Example Test Configuration View for processes

The diagram in Figure 17 shows another Test Configuration View, this time based on a number of Processes that must be tested. Each Process is represented by a SysML block and these Processes will also appear as life lines on corresponding Test Case Behaviour Views.

3.5.3 Example Test Behaviour Viewpoint

The ‘Test Behaviour View’ may be visualised using a number of SysML behaviour diagrams.

- Sequence diagrams. Used for defining scenarios and high-level testing, such as acceptance testing, integration testing, performance testing, etc.
- State machine diagrams. Used for state-based testing, such as unit testing, functional testing, mode-based testing, etc.
The Test Pattern

- Activity diagrams. Used for detailed testing, such as unit testing, functional; testing, process-based testing, etc.
- Parametric diagrams. Used for defining mathematical-based scenarios for testing at any level.

The diagram in Figure 18 shows an example of a Test Case Behaviour View where a sequence diagram is used to show the interactions between Stakeholder Roles in the execution of a specific Test Case.

In the example shown here, the Scenario focuses on the interactions between the various Stakeholder Roles and the order of these interactions. So in order to perform this Test Case, the first thing that happens is that the ‘Escapologist’ has a self-interaction which is to ‘get in coffin’. The next thing that happens is that the ‘Assistant’ closes the lid, and so on.

It is possible to use this View to capture not only what constitutes a successful Test Case but also how the System must be tested to respond to some sort of problem, as shown in the next example.
The Test Pattern

The diagram in Figure 19 shows another example Test Case Behaviour View but, this time, the Test Case reflects what must happen when a problem occurs during the execution of the System.

The example here shows how the Processes from Figure 17 are executed and, therefore, how they must be tested. In this case the ‘Set up’ Processes is executed and then the ‘Start’ Process is executed which simultaneously sends out two messages ‘start escape’ to the ‘Escape’ Process and ‘monitor’ to the ‘Monitor’ Process. In this example, the ‘<Monitor’ Process detects a ‘time out’ and invokes the ‘Emergency’ Process.

3.5.4 Example Test Record Viewpoint

The Test Record View captures any test results that are generated during the execution of a Test Case.
The diagram in Figure 20 shows an example of how test results may be captured and then annotated to a copy of the original Test Behaviour View. In this example, the Test Case was executed successfully and a simple SysML note has been added to the copy in order to record the results.

Figure 21 Example Test Record View for unsuccessful Test Case execution

The Test Pattern
4 Summary

The Testing Pattern provides a template for testing that can be applied to almost any aspect of a System Model.

This Pattern may also be used as a basis for automating the tests using model-based testing techniques using an MBSE tool.

This Pattern also lends itself to non-SysML modelling such as the use of formal methods and simulation techniques.

5 Related Patterns

If using the Interface Definition Pattern, the following Patterns may also be of use:

- Description
- Traceability

6 References & Further Reading


