Test Process Improvement for Automated Test Generation

Version: 1.01.51 (2010-05-06) [Draft]

TPI is an evaluation framework to evaluate the maturity of a software testing process. We have modified the original TPI framework to evaluate test processes that are using Automated Test Generation methods and tools.

Henri Heiskanen, Mika Maunumaa, Mika Katara
10.2.2010

Tampere University of Technology, Department of Software Systems
# Table of Contents

Table of Contents ................................................................................................................................. 2

1 Introduction.................................................................................................................................. 3

2 Background .................................................................................................................................. 4

   2.1 Test Process Improvement (TPI) ........................................................................................... 4
   2.2 Automated Test Generation (ATG) ....................................................................................... 5
   2.3 Domain Specific Testing ....................................................................................................... 6
   2.4 Models and modeling ............................................................................................................. 6

3 Introduction to TPI for Automated Test Generation .................................................................... 7

   3.1 TPI Cornerstones and key area .............................................................................................. 7
   3.2 Maturity Matrix for ATG-TPI ............................................................................................... 7

4 Dependencies ............................................................................................................................. 11

5 Key areas .................................................................................................................................... 12

   5.1 Life-Cycle ............................................................................................................................ 12
   5.2 Techniques ........................................................................................................................... 15
   5.3 Organization.......................................................................................................................... 16
   5.4 Infrastructure ......................................................................................................................... 18
   5.5 General................................................................................................................................. 24
   5.6 Automated Test Generation ................................................................................................. 26

6 Baseline Maturity Profile for Automated Test Generation ........................................................ 31

7 Future of Test Process Improvement - TPI NEXT .................................................................... 32

   7.2 Other updates ....................................................................................................................... 32
   7.3 Something new ..................................................................................................................... 33

8 Glossary ..................................................................................................................................... 34

9 Bibliography............................................................................................................................... 36
1 Introduction

The software testing industry is slowly transitioning from manually created test cases to automatically generated ones, which can be executed either after the generation phase (offline testing) or during the generation phase (online testing). While manual testing will always have a role in GUI sensitive systems, this change calls for rethinking the actual testing process. New roles and tasks must be introduced into the testing organization. In addition, parts of the software development life cycle other than testing must be adjusted as well to accommodate the new, Automated Test Generation (ATG) practice. The new way of working will affect many parts of an organization, but the big question is whether the organization is mature enough to adapt to and sustain the new testing approach.

In this document we will outline a Test Process Improvement (TPI for short) [1] add-on for automated test generation. The aim of the work is to provide support for the assessment of a testing organization in terms of how mature it is when it comes to using automated test generation practices and how the organization can evolve in this field of testing. Another aim is to provide a baseline TPI profile in order for an organization to start using the ATG practice. The profile outlines the minimum requirements for a successful ATG-oriented testing organization.

The ATG-TPI introduces 4 new key areas that are assessed. We think that the way how the basis of automated test generation is created and how that basis is applied to testing are relevant factors for process maturity. Equally important is how confident one can be in the quality of the system after testing. This does not suggest that there are no errors in the test object, but the test object has reached the required quality level. Since the area of ATG involves a wide range of software engineering and computer science skills in addition to testing skills, we also think that the evaluation of technological and methodological knowledge is very important for the successful adaptation of automated test generation practices.
2 Background

The TPI presumes the use of conventional software testing practices and traditional test automation techniques (if opted for). As it is, this is no longer the case since the arrival of technologies such as automated test generation. What this change toward ATG practices means in practical terms is the fact that an organization willing to apply the ATG approach must achieve a certain level of maturity in order for the introduction of ATG to be successful. The baseline TPI profile for an ATG-oriented organization is provided later on in this document.

In this chapter we will explain some background information about TPI and automated test generation.

2.1 Test Process Improvement (TPI)

No matter how well the testing is performed, there is always room for improvement. To improve, the current situation needs to be identified – and that can be done with TPI. TPI helps to identify the current situation and suggests actions for improving your testing process. TPI also provides a frame of reference to identify the strengths and weaknesses of the process. The method is based on the structured test approach called TMap [3], which was developed in a Dutch company called IQUIP in 1995. Based on the experiences using the TMap the company and their clients came to the conclusion that they need a method that would support their improvement efforts. The development of TPI began in 1996 and its final version was introduced in 1998.

Figure 1: Test Process Improvement model [1]

TPI is based on four cornerstones (Figure 1) that depict various important aspects of the software development process, namely life cycle, techniques, infrastructures and tools, and organization. In addition, there is a general category for other items. The four cornerstones contain several key areas, which group various aspects within each cornerstone. Each key area has several levels of
maturity, on a scale of one to four, and each maturity level contains several checkpoints that specify the steps to take in order to reach the desired maturity level in practical terms.

2.2 Automated Test Generation (ATG)

Conventional test automation relies on scripts written in different languages and by different people. Sometimes script authors have formal software engineering backgrounds, but often that is not the case. Writing a test script that verifies functionality thoroughly or at least adequately can be a tiresome task. Often there are dozens of scripts to test just one feature adequately. A test script must be small and simple to keep it maintainable. Otherwise the script will face the same problems the code it is supposed to test. To circumvent these problems, automatic test generation can be used to generate tests automatically, both before and after the completion of the test object.

The generation of test cases always requires some kind of source from which the tests are generated. The source itself can be varied, for example, requirements, design models, user documents (which in TPI are collectively called the test basis) and/or conventional test scripts. We call those an ATG basis. The key issue here is that they are expressed in a form that is computable, i.e., they can be handled with a computer. If they are not in a computational form they have to be translated to such. The result from that translation is called the test model.

In practice, the test model can be one monolithic model or it can comprise several component models specified with different kinds of modeling formalism. The formalism depends on the perspective by which the system is modeled and the model can in turn contain information from one or many perspectives, for instance, system/design and testing perspectives. One property that is very important for models is that there is a description for both functionality (control-flow) and data (input/output values). The functionality is often modeled using some form of state machine and the data is often a simple data table, but the data can also be incorporated into a state machine, for instance in the case of state charts.

Depending on the nature of the test object and available tools, tests can be generated before test execution (offline) or during test execution (online). Both these approaches have their advantages and disadvantages but there are some organizations and systems in which one way works better than the other. The big difference is that offline generation can produce huge numbers of conventional test cases, which can be executed and measured using conventional ways. And since they are close to conventional test cases people can “understand” and accept them easier than the outcome of online testing. In online testing, the test execution can be arbitrarily long and its content can vary between consecutive executions. Thus, one execution can contain various “test cases” in conventional terms. The actual trace (execution path) depends on what opportunities the model provides to be acted upon and what kind of feedback the test object gives in response to test steps.

The most intimidating thing in automated test generation is the model and act of modeling. As already explained, a model is a representation of the information contained by the ATG basis with a certain level of abstraction. There can be an explicit model, which means that it is explicitly modeled by a modeler, or an implicit model, which is more like a mental model and is often conceived during the execution of a test. An example of a mental model is a random monkey test for a GUI, where the automation system identifies items on the screen and issues write-text and press-button commands in a random order.
Even though both models have their uses, the explicit model is more beneficial since it makes it possible to verify what state the test object should be in. However, finding the current state of the test object is often the most challenging task. The state verification becomes even harder as the testing level rises toward that of system testing. And on top of that, the modeling process itself can be tedious and time consuming even for an experienced modeler.

### 2.3 Domain Specific Testing

A general scope is often way too broad to handle effectively and efficiently, because it tries to manage everything or solve every problem. For example, GUI testing is generally a large area since there are various kinds of GUIs on different platforms. If one tries to automate testing in all of those various platforms, it will be a “mission impossible” because there are so many ways to do the same thing. A more efficient solution is to narrow down the area one is working on and solve problems within that given area. This condensed area is called a domain or problem domain.

The concept of domain narrows down the application area in a controlled way. For example, one can work on GUI testing in a desktop or mobile domain. To further narrow down the domain, it can be specified as a Windows Mobile domain or S60 domain. If the organization develops CAD tools, the domain will be defined as a “Computer Aided Design tools” domain.

So why is this domain concept important? The domain specifies the vocabulary and way of communicating within it. It specifies a language that contains words (actions that can be issued in the domain), the syntax (how actions are related and what kind of information they can convey), and the semantics (when a particular action can be issued and what it expects in response). A domain-specific language can be textual, but often it is more useful as a graphical language. Such a domain-specific language for a GUI can contain actions for various graphical elements. However, it may be better to raise the level of abstraction from technical level to user level. When the domain-specific language is at the user level, it comes closer to the user and that makes it easier to comprehend than the technical engineering domain, which may not be related to the user’s daily work.

Domain-specific testing means that the testing is focused at the level where a particular aspect of a system has been validated using the tools and test strategies that are suitable for the given testing level and appropriate for the given domain.

### 2.4 Models and modeling

As already discussed, there can be different sorts of models created from different perspectives, for instance system/design models and test models. System/design models are created during system development and they present an internal view of the system as it is supposed to be. On the other hand, test models, which represent the external view, are created for testing purposes with a destructive attitude that is suitable for testing; they can be seen as a “second opinion” on requirements. From the process point of view, it makes sense to reuse design artifacts as much as possible. On balance, we need a “second opinion” to make sure that the requirements have been correctly implemented.
3 Introduction to TPI for Automated Test Generation

TPI has 20 key areas that cover the general testing well. However the Automated Test Generation approach has some particularities that need to be addressed when assessing the testing process. Here we will give an overview of TPI and the four new key areas.

3.1 TPI Cornerstones and key area

The original TPI comprises four cornerstones, which in turn consist of various key areas. The cornerstones are:

- Life Cycle,
- Infrastructure and Tools,
- Techniques,
- Organization.

In addition, there is a broad category “All” under which two key areas fall. The cornerstone of Life Cycle contains key areas for Test Strategy, Life-Cycle Model and Moment of Involvement. The cornerstone of infrastructure and tools has four key areas: Test Automation, Testing Environment and Office Environment. The cornerstone of Techniques includes key areas for Metrics, Static Testing, Test Specification Techniques and Estimating and Planning. Finally, the cornerstone of Organization holds following key areas: Commitment and Motivation, Test Functions and Training, Scope of Methodology, Communication, Reporting, Defect Management, Test Process Management and Testware Management. Key areas of Evaluation and Low-Level Testing are placed under the general “All” category. More specific descriptions of the key areas are given later on.

3.2 Maturity Matrix for ATG-TPI

The maturity matrix outlines all key areas and their respective maturity levels from A to D. Each maturity level has a short description as to what kind of maturity it represents in its corresponding key area. The maturity matrix for automated test generation is depicted in Table 1. Key areas that belong to the same cornerstone have common coloring. The changes introduced by the ATG modification are indicated by green color. New key areas are located at the end of the table. New maturity levels are indicated with dark green and altered original maturity levels by a light green color.
<table>
<thead>
<tr>
<th>Maturity level</th>
<th>Key area</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Life cycle</td>
<td>Test strategy</td>
<td>Strategy for single high-level test</td>
<td>Combined strategy for high-level tests</td>
<td>Combined strategy for all test and evaluation levels</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Life cycle model</td>
<td>Planning, Specification, Execution</td>
<td>Planning, Preparation, Specification, Execution and Completion</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Moment of involvement</td>
<td>Completion of test basis</td>
<td>Start of test basis</td>
<td>Start of requirements definition</td>
<td>Project initiation</td>
</tr>
<tr>
<td></td>
<td>Estimating and planning</td>
<td>Substantiated estimating and planning</td>
<td>Statistically substantiated estimating and planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Test specification</td>
<td>Informal techniques</td>
<td>Check-lists</td>
<td>Modeling of test basis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Static test techniques</td>
<td>Inspection of test basis</td>
<td>Check-lists</td>
<td>Modeling of test basis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Metrics</td>
<td>Project metrics (product)</td>
<td>Project metrics (process)</td>
<td>System metrics</td>
<td>Organization metrics (&gt;1 system)</td>
</tr>
<tr>
<td></td>
<td>Test automation</td>
<td>Use of tools</td>
<td>Managed test automation</td>
<td>Optimal test automation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Test environment</td>
<td>Managed and controlled test environment</td>
<td>Testing in the most suitable environment</td>
<td>“Environment-on-call”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Office environment</td>
<td>Adequate and timely office environment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Commitment and motivation</td>
<td>Assignment of budget and time</td>
<td>Testing integrated in project organization</td>
<td>Test-engineering</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Test functions and training</td>
<td>Test manager, modeler and testers</td>
<td>(Formal) Methodical, Technical and Functional Support, Management</td>
<td>Formal internal Quality Assurance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scope of methodology</td>
<td>Project specific</td>
<td>Organization generic</td>
<td>Organization optimizing, R&amp;D activities</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Communication</td>
<td>Internal communication</td>
<td>Project communication (defects, change control)</td>
<td>Communication in organization about the quality of the test processes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reporting</td>
<td>Defects</td>
<td>Progress (status of tests and products), activities (costs and time, milestones), defects with priorities</td>
<td>Risks and recommendations, substantiated with metrics</td>
<td>Recommendations have a Software Process Improvement character</td>
</tr>
<tr>
<td></td>
<td>Defect management</td>
<td>Internal defect management</td>
<td>Extensive defect management with flexible reporting facilities</td>
<td>Project defect management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Testware management</td>
<td>Internal testware management</td>
<td>External management of test basis and test object</td>
<td>Reusable testware</td>
<td>Traceability system requirements to test cases</td>
</tr>
<tr>
<td></td>
<td>Test process management</td>
<td>Planning and execution</td>
<td>Planning, execution, monitoring and adjusting</td>
<td>Monitoring and adjusting in organization</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Evaluation</td>
<td>Evaluation techniques</td>
<td>Evaluation strategy</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Low-level testing</td>
<td>Low-level test life cycle model (planning, specification and execution)</td>
<td>White-box techniques</td>
<td>Low-level test strategy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modeling approach</td>
<td>Monolithic test model</td>
<td>Abstract and domain-specific test model</td>
<td>(Re-)Use of design models</td>
<td>Test modeling integrated to design modeling</td>
</tr>
<tr>
<td></td>
<td>Use of models</td>
<td>Input generation with context knowledge</td>
<td>Input generation with domain knowledge</td>
<td>Test object output verification</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Test confidence</td>
<td>Critical (path) functionality confidence</td>
<td>Shallow functionality confidence</td>
<td>Thorough functional confidence</td>
<td>Non-functional requirements covered</td>
</tr>
</tbody>
</table>
In Table 2 all maturities of various key areas are placed so that their relative importance can be seen. There are three improvement categories where all maturities are placed: Controlled, Efficient, and Optimized.

Table 2: Maturity categories

<table>
<thead>
<tr>
<th>Key Area</th>
<th>Scale</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Strategy</td>
<td>Initial</td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Life Cycle Model</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moment of Involvement</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estimating and Planning</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Specification Techniques</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static Test Techniques</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metrics</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Automation</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Environment</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Office Environment</td>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commitment and Motivation</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Functions and Training</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scope of Methodology</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reporting</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defect Management</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testware Management</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Process Management</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluation</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low-level Testing</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modeling Approach</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use of Models</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Confidence</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technological and Methodological Knowledge</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A test process that is within the controlled category provides enough insight into the quality of the test object. A test process within efficient category is well integrated into the whole software.
development organization. When a test process is optimizing it will continuously evaluate its performance and tries to adapt its efficiency to needs for today.

Using the maturity matrix above the organization can determine where they are now and where they want to be. The organization can choose for example that their testing process should be totally within controlled level or they can decide that some parts are controlled and other will be efficient.
4 Dependencies

Several maturity levels of various key areas depend on other maturity levels of other key areas. Many TPI Key areas are related this way, thus connecting different maturity levels to each other. In order for an organization to reach a certain level of maturity (e.g. KA 3: Moment of Involvement level A), the organization must also have the required level of maturity in other key areas (in this case, KA 2: Life cycle model level A).

Table 3: Dependency matrix for Key Area maturities (x = has dependency, o--o = conditional dependency)

<table>
<thead>
<tr>
<th>Key area</th>
<th>Maturity level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test strategy</td>
<td>A</td>
</tr>
<tr>
<td>Life cycle model</td>
<td>A</td>
</tr>
<tr>
<td>Moment of involvement</td>
<td>A</td>
</tr>
<tr>
<td>Planning and reviewing</td>
<td>A</td>
</tr>
<tr>
<td>Specifications techniques</td>
<td>A</td>
</tr>
<tr>
<td>Stake test techniques</td>
<td>A</td>
</tr>
<tr>
<td>Metrics</td>
<td>A</td>
</tr>
<tr>
<td>Test automation</td>
<td>A</td>
</tr>
<tr>
<td>Test environment</td>
<td>A</td>
</tr>
<tr>
<td>Use of models</td>
<td>A</td>
</tr>
<tr>
<td>Test assurance</td>
<td>A</td>
</tr>
<tr>
<td>Test functions and training</td>
<td>A</td>
</tr>
<tr>
<td>Communication</td>
<td>A</td>
</tr>
<tr>
<td>Estimating and planning</td>
<td>A</td>
</tr>
<tr>
<td>Commitment and motivation</td>
<td>A</td>
</tr>
<tr>
<td>Test process management</td>
<td>A</td>
</tr>
<tr>
<td>Defect management</td>
<td>A</td>
</tr>
<tr>
<td>Testware management</td>
<td>A</td>
</tr>
<tr>
<td>Use of models</td>
<td>A</td>
</tr>
<tr>
<td>Test confidence</td>
<td>A</td>
</tr>
<tr>
<td>Use of models</td>
<td>A</td>
</tr>
<tr>
<td>Test strategy</td>
<td>A</td>
</tr>
<tr>
<td>Life cycle model</td>
<td>A</td>
</tr>
<tr>
<td>Moment of involvement</td>
<td>A</td>
</tr>
</tbody>
</table>

Table 3 presents the dependencies between the key areas. In the left column is the key area whose dependencies on other key areas is examined, and the uppermost row lists the other key areas that any key area might be dependent on. A dependency of a certain maturity level can be determined as follows: at first the key area is selected from the column on the left as well as the corresponding maturity level row. By following the row one can determine if there are any letters on the row. Those letters corresponds the maturity level on which the selected maturity level is depending on. By following the column up one can found the corresponding key area.
5 Key areas

This chapter introduces the key areas and their respective checkpoints. The additions for the ATG approach are also presented.

The changes introduced for ATG are highlighted in bold in the checkpoint list; all other information in the checkpoint lists is from the original TPI.

5.1 Life-Cycle

In following all changed to key areas of Life-Cycle are introduced.

5.1.1 KA01 Test Strategy

“The test strategy has to be focused on detecting the most important defects as early and as economically as possible. The test strategy defines which requirements and (quality) risks are covered by what tests. The better each test level defines its own strategy and the more the different test level strategies are adjusted to each other, the higher the quality of the overall test strategy.”

[1]

No changes introduced in ATG-TPI.
5.1.2 KA02 Life-cycle Model

“Within the test process a number of phases can be defined, such as planning, preparation, specification, execution and completion. In each phase several activities are performed. For each activity the following aspects should be defined: purpose, input, process, output, dependencies, applicable techniques and tools, required facilities, documentation, etc. The importance of using a life-cycle model is an improved predictability and controllability of the test process, because the different activities can be planned and monitored in mutual cohesion.” [1]

5.1.2.A (A) Planning, Specification, Execution

**Checkpoints**

1. For the test (at least) the following phases are recognized: planning, test specification, and test execution. These are subsequently performed, possibly per subsystem. A certain overlap between the phases is allowed.

2. Activities to be performed per phase are:
   
a. Formulate assignment, determine the test basis, determine test strategy, set up organization, set up test deliverables, define infrastructure and tools, set up management, determine planning, produce test plan (phase Planning);

b. Design test cases, specify intake of test object (test data and input) and infrastructure, realize test infrastructure (phase Specification);

c. Take in test object and infrastructure, set up starting test databases, execute (re)tests (phase Execution).

d. With ATG, also create a test model from the test basis and in addition to designing test cases, prepare test generation infrastructure (phase Specification);

**Improvement suggestions**

Test execution is often in critical path when the development releases the system for testing. That is why it is important to start test modeling before that release, because the modeling can apply test basis and it is not dependent on test object. Also the adaptation of the test generation infrastructure to the forth coming test object should be started as early as possible.

5.1.3 KA03 Moment of involvement

“Although the actual execution of the test normally begins after the realization of the software, the test process must and can start much earlier. An earlier involvement of testing in the system development path helps to find defects as soon and easy as possible and perhaps even to prevent errors. A better adjustment between the different tests can be done and the time that testing is on the critical path of the project can be kept as short as possible.” [1]
5.1.3.A  (A) Completion of test basis

**Checkpoints**

1. The activity "testing" starts simultaneously with or earlier than the completion of the test basis for a restricted part of the system that is to be tested separately.

2. (The system can be divided into several parts which are finished, built and tested separately. The testing of the first subsystem has to start at the same time or earlier than the completion of the test basis of that particular subsystem.)

3. **Modeling of test basis has been started (low-level models may be created to aid this process)**

5.1.3.B  (B) Start of test basis

**Checkpoints**

1. The activity "testing" starts simultaneously with or earlier than the phase in which the test basis (often the functional specification) is defined.

2. **Modeling of functionality has been started (abstract models may be created to aid this process)**

5.1.3.C  (C) Start of Requirements Definition

**Checkpoints**

1. The activity "testing" starts simultaneously with or earlier than the phase in which the requirements are defined.

2. **Modeling of abstract requirements (very shallow and abstract models may be created at this point)**

5.1.3.D  (D) Project initiation

No changes introduced for ATG-TPI.
5.2 Techniques
In following all changed to key areas of Techniques are introduced.

5.2.1 KA04 Estimating and Planning
“Test planning and estimating indicate which activities have to be carried out when, and the necessary resources (people). Good estimating and planning are very important, because they are the basis of, for example, allocating resources for a certain time frame.” [1]

No changes introduced for ATG-TPI.

5.2.2 KA05 Test Specification Techniques
“The definition of a test specification technique is "a standardized way of deriving test cases from source information". Applying these techniques gives insight into the quality and depth of the tests and increases the re-usability of the test.” [1]

An additional maturity level was added to better accommodate the needs of ATG.

5.2.2.A (A) Informal techniques
No changes introduced for ATG-TPI.

5.2.2.B (B) Formal techniques
No changes introduced for ATG-TPI.

5.2.2.C (C) Computational techniques

### Checkpoints

1. Test specifications can be derived from a computational model.

5.2.3 KA06 Static Testing
“Not everything can and should be tested dynamically, that is, by running programs. Inspection of products without running programs, or the evaluation of measures which must lead to a certain quality level, is called static tests. Checklists are very useful for this.” [1]

An additional maturity level was added to better accommodate the needs of ATG.
5.2.3.A (A) Inspection of test basis
No changes introduced for ATG-TPI.

5.2.3.B (B) Checklists
No changes introduced for ATG-TPI.

5.2.3.C (C) Modeling of test basis

Checkpoints

1. A test model of the test basis is created.
2. Test modeling is used to validate features defined by the test basis.

5.2.4 KA07 Metrics

“Metrics are quantified observations of the characteristics of a product or process. For the test process, metrics of the progress of the process and the quality of the tested system are very important. They are used to control the test process, to substantiate the test advice and also to make it possible to compare systems or processes. Why has one system far fewer failures in operation than another system or why is one test process faster and more thorough than another? Specifically for improving the test process, metrics are important by evaluating consequences of certain improvement actions, by comparing data before and after performing the action.” [1]

No changes introduced for ATG-TPI.

5.3 Organization

In following all changed to key areas of Organization are introduced.

5.3.1 KA08 Test automation

“Automation within the test process can take place in many ways and has in general one or more of the following aims:

- fewer hours needed,
- shorter lead time,
- more test depth,
- increased test flexibility,
- more and/or faster insight in test process status,
- better motivation of the testers.” [1]
The changes introduced for ATG are highlighted in bold in the checkpoint list of each maturity level:

5.3.1.A  (A) Use of Tools

Checkpoints

1. A decision has been taken to automate certain activities in the planning, specification and/or execution phases. The test management and the party who pays for the investment in the tools (generally the line management or project management) are involved in this decision;

2. Use is made of automated tools that support certain activities in the planning, specification and execution phases (such as a scheduling tool, a defects registration tool and/or home-built stubs and drivers);

3. The test management and the party paying for the investment in the tools acknowledge that the tools being used provide more advantages than disadvantages.

4. There is a suitable tool to model the ATG basis.

5.3.1.B  (B) Managed Test Automation

Checkpoints

1. A well-considered decision has been taken regarding the parts of test specification and execution that should or should not be automated. This decision involves those types of test tools and test activities that belong to the test execution.

2. If the decision on automation of the test execution is a positive one, there will now also be a tool for test execution.

3. The introduction of new test tools is preceded by an inventory of technical aspects (does the test tool work in the infrastructure?) and any possible preconditions set for the test process (for example, test cases should be established in a certain structure instead of in a free-text form, so that the test tool can use this as input);

4. If use is made of a Capture & Playback tool for automated test execution, explicit consideration should be given during implementation to maintainability of the test scripts included;

5. Most of the test tools can be reused for a subsequent test process. To do so, the management of the test tools has been arranged. The fact that ‘in general’ test tools should be reusable, means that the test tools that are used explicitly within one test process need not be reusable; The use of the test tools matches the desired methodology of the test
process, which means that use of a test tool will not result in inefficiency or undesired limitations of the test process.

6. Models are reusable within the domain.

5.3.1.C (C) Optimal Test Automation

No changes introduced for ATG-TPI.

5.3.2 KA09 Testing Environment

“The test execution takes place in a so-called test environment. This environment mainly comprises the following components:

- hardware;
- software;
- means of communication;
- facilities for building and using databases and files;
- procedures.

The environment should be composed and set up in such a way that by means of the test results it can be optimally determined to what extent the test object meets the requirements. The environment has a large influence on the quality, lead time, and cost of the test process. Important aspects of the environment are responsibilities, management, on-time and sufficient availability, representativeness, and flexibility.” [1]

No changes introduced for ATG-TPI.

5.3.3 KA10 Office Environment

“The test staff needs rooms, desks, chairs, PCs, word-processing facilities, printers, telephones, and so on. A good and timely organization of the office environment has a positive influence on the motivation of the test staff, on communication in- and outside the team, and on the efficiency of the work.” [1]

No changes introduced for ATG-TPI.

5.4 Infrastructure

In following all changed to key areas of Infrastructure are introduced.
5.4.1 KA11 Commitment and Motivation

“The commitment and the motivation of the persons involved in testing are important prerequisites for a smoothly running test process. The persons involved are not only the testers, but also, for example, the project management and the line management personnel. The latter are mainly important in the sense of creating good conditions. The test process thus receives enough time, money, and resources (quantitatively and qualitatively) to perform a good test, in which cooperation and good communication with the rest of the project results in a total process with optimum efficiency.” [1]

No changes introduced for ATG-TPI.

5.4.2 KA12 Test Functions and Training

“In a test process the correct composition of a test team is very important. A mix of different disciplines, functions, knowledge, and skills is required. Besides specific test expertise, knowledge of the subject matter, knowledge of the organization and general IT knowledge is required. Social skills are also important. For acquiring this mix, training etc. is required.” [1]

The changes introduced for ATG are highlighted in bold in the checkpoint list of each maturity level:

5.4.2.A (A) Test manager, testers, modelers, and test environment specialist

<table>
<thead>
<tr>
<th>Checkpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The test personnel consist at the very least of a test manager, a number of testers, modeler(s), and test environment specialist.</td>
</tr>
<tr>
<td>2. The tasks and responsibilities have been defined.</td>
</tr>
<tr>
<td>3. The test personnel has had specific test training (e.g. test management, test techniques, etc.) or has sufficient experience in the field of testing.</td>
</tr>
<tr>
<td>4. For the acceptance test, expertise in the subject matter is available to the test team.</td>
</tr>
<tr>
<td>5. Tasks and roles for modeler and test environment specialist have been defined.</td>
</tr>
</tbody>
</table>

5.4.2.B (B) (Formal) Methodical, Technical and Functional Support, Management of the test process, testware and infrastructure

<table>
<thead>
<tr>
<th>Checkpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
1. The task Methodical Support is outlined separately. Its activities are defining and maintaining test instructions, procedures and techniques and advising about and evaluating the right application of the above. The **modeling methodology is also outlined as part of this process.**

2. The task Technical Support is outlined separately.

3. The task Functional Support is outlined separately.

4. The task Management test process is outlined separately and is responsible for the registration, storage, and availability of all management objects of the test process. Sometimes one will carry out the management oneself, in other cases one will organize and/or evaluate that management. Objects to be managed are progress, budgets, and defects.

5. The task Management testware is outlined separately and is responsible for the registration, storage, and availability of all management objects of the testware. Sometimes one will carry out the management oneself, in other cases one will organize and/or evaluate that management. Objects to be managed are test documentation, test basis, test objects (internal), test cases including test files and databases, test instructions and procedures.

6. The task Management test infrastructure is outlined separately and is responsible for the registration, storage, and availability of all management objects of the test infrastructure. Sometimes one will carry out the management oneself, in other cases one will organize and/or evaluate that management. Objects to be managed are test environments (test databases) and test tools.

7. The persons who carry out these tasks have sufficient knowledge and experience.

8. The time needed for these tasks is planned. Supervision is carried out to see that these tasks are in fact performed.

**5.4.2.C (C) Formal internal Quality assurance**

No changes introduced for ATG-TPI.

**5.4.3 KA13 Scope of Methodology**

“For each test process in the organization a certain methodology or working method is used, comprising activities, procedures, regulations, techniques etc.. When these methodologies are different each time or when the methodology is so generic that many parts have to be drawn up again each time, it has a negative effect on the test process efficiency. The aim is that the organization uses a methodology which is sufficiently generic to be applicable in every situation, but which contains enough detail so that it is not necessary to rethink the same items again each time.” [1]

No changes introduced for ATG-TPI.
5.4.4 KA14 Communication

“In a test process, communication with the people involved must take place in several ways, within the test team as well as with parties such as the developer, the user, the customer, etc.. These communication forms are important for a smoothly running test process, not only to create good conditions and to optimize the test strategy, but also to communicate about the progress and the quality.” [1]

The changes introduced for ATG are highlighted in bold in the checkpoint list of each maturity level:

5.4.4.A  (A) Internal communication
No changes introduced for ATG-TPI.

5.4.4.B  (B) Project communication (defects, change control)

Checkpoints

1. In the test team meeting minutes are taken.

2. In the test team meeting, besides progress and the quality of the test object, the quality of the test process is a fixed subject on the agenda.

3. Periodically, the test manager reports progress and the quality of the object to be tested, including the risks, in the project meeting. The test manager also reports the quality of the test process.

4. Agreements in this meeting are documented.

5. The test manager is informed in time about changes in planned and agreed delivery dates (test basis as well as object).

6. In a periodic defects meeting (or analysis meeting) solutions to defects are discussed between representatives of the test team and of other parties involved.

7. Testing is involved in change control for judging the impact of change proposals on the test effort.

8. Modeling is involved in change control for judging the impact of change proposals on the modeling effort.

5.4.4.C  (C) Communication within the organization about the quality of the test process
No changes introduced for ATG-TPI.
5.4.5 KA15 Reporting

“Testing is not so much "defect detection" as about giving insight in the quality level of the product. Reporting should be aimed at giving well-founded advice to the customer concerning the product and even the system development process.” [1]

No changes introduced for ATG-TPI.

5.4.6 KA16 Defect Management

“Although managing defects is in fact a project matter and not specifically of the testers, the testers are mainly involved in it. Good management should be able to track the life-cycle of a defect and also to support the analysis of quality trends in the detected defects. Such analysis is used, for example, to give well-founded quality advice.” [1]

The changes introduced for ATG are highlighted in bold in the checkpoint list of each maturity level:

5.4.6.A  (A) Internal defect management
No changes introduced for ATG-TPI.

5.4.6.B  (B) Extensive defect management with flexible reporting facilities

<table>
<thead>
<tr>
<th>Checkpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Defect data needed for later trend analysis recorded in detail:</td>
</tr>
<tr>
<td>1. test case</td>
</tr>
<tr>
<td>2. test</td>
</tr>
<tr>
<td>3. subsystem</td>
</tr>
<tr>
<td>4. priority (blocking Y/N)</td>
</tr>
<tr>
<td>5. program plus version</td>
</tr>
<tr>
<td>6. test basis plus version</td>
</tr>
<tr>
<td>7. cause (probable + definitive)</td>
</tr>
<tr>
<td>8. all status transitions of the defect including dates</td>
</tr>
<tr>
<td>9. a description of the problem solution</td>
</tr>
<tr>
<td>10. version of the test object in which the defect is solved</td>
</tr>
</tbody>
</table>
11. problem solver

2. The administration lends itself for extensive reporting possibilities, which means that reports can be selected and sorted in different ways.

3. There is someone responsible for ensuring that defect administration is carried out properly and consistently.

4. **Error trace (shortened to a minimum, if possible) is reported in addition to 16.B.1**
   
   1. test configuration
   
   2. test model and its version

---

5.4.6.C **(C) Project defect management**

No changes introduced for ATG-TPI.

5.4.7 **KA17 Testware Management**

“The products of testing should be maintainable and reusable and so they must be managed. Besides the products of the testing themselves, such as test plans, specifications, databases and files, it is important that the products of previous processes such as functional design and realization are managed well, because the test process can be disrupted if the wrong program versions, etc. are delivered. If testers make demands upon version management of these products, a positive influence is exerted and the testability of the product is increased.” [1]

The changes introduced for ATG are highlighted in bold in the checkpoint list of each maturity level:

5.4.7.A **(A) Internal testware management**

No changes introduced for ATG-TPI.

5.4.7.B **(B) External management of test basis and test object**

<table>
<thead>
<tr>
<th>Checkpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The test basis and the test object (usually design and software) are managed by the project according to a described procedure, with steps for delivery, registering, archiving and reference.</td>
</tr>
<tr>
<td>2. Management contains the relationships between the various parts (test basis and test object).</td>
</tr>
<tr>
<td>3. The test team is informed about changes in test basis or test object in a timely fashion.</td>
</tr>
<tr>
<td>4. The modeler is informed about changes in the test basis and test object in a timely fashion.</td>
</tr>
</tbody>
</table>
5.4.7.C  (C) Reusable testware
No changes introduced for ATG-TPI.

5.4.7.D  (D) Traceability system requirements to test cases

Checkpoints

1. Each system requirement and specification is related to one or more test cases in a transparent way, and vice versa.

2. These relations are traceable through separate versions (e.g. system requirement A, version 1.0, is related to functional design B, version 1.3, is related to programs C and D, version 2.5 and 2.7, and is related to test cases X to Z, version 1.4).

3. Requirements can be traced to test models and vice versa.

5.4.8 KA18 Test Process Management

“For managing each process and activity, the four steps from the Deming circle are essential: plan, do, check and act. Process management is of vital importance for the realization of an optimal test in an often turbulent test process.” [1]

No changes introduced for ATG-TPI.

5.5 General

In following all changed to key areas of general category are introduced.

5.5.1 KA19 Evaluation

“Evaluation means inspecting intermediate products such as the requirements and the functional design. The importance of evaluation is that the defects are found at a much earlier stage in the development process than with testing. This makes the rework costs much lower. Also, evaluation can be set up more easily because there is no need to run programs or to set up an environment etc.” [1]

The changes introduced for ATG are highlighted in bold in the checkpoint list of each maturity level:

5.5.1.A  (A) Evaluation techniques
1. In evaluating (intermediate) products techniques are used, in other words a formal and described working method is applied.

2. Reporting of the evaluation and its results takes place.

3. The handling of the results is monitored.

4. Testers are involved in these evaluations.

5. Modelers are involved in these evaluations.

5.5.1.B (B) Evaluation strategy

No changes introduced for ATG-TPI.

5.5.2 KA20 Low-level Testing

“The low-level tests are almost exclusively carried out by the developers. Well-known low-level tests are the unit test and the integration test. Just as evaluation, the tests find defects at an earlier stage of the system development path than the high-level tests. Low-level testing is efficient, because it requires little communication and because often the finder is both the error producer as well as the one who corrects the defect.” [1]

There are various options how ATG is applied for low-level testing. Since low-level tests can yield dozens of test cases per unit containing often 3-5 times the amount of code in the unit under test it will result maintenance problem. Thus it would be wise to use approaches that consider the unit under test as an implicit model and creates basic set of test cases out of that code. Although it is a little problematic to use code as a model from testing perspective, it will create enough test cases to verify the current implementation. In addition to that there is a place for specification based test cases, but often in real world those must be created by hand by tester or developer.

5.5.2.B (B) White-box techniques

4. An ATG-based tool is used to generate low-level tests.

Approach that is used to generate low-level tests uses either static or dynamic analysis or both to determine test cases required.

5.5.2.C (C) Low-level test strategy

6. Well-informed decision has been made to use ATG-based tool for low-level testing as a part of the test strategy.

The decision whether to choose ATG tool should be rationalized with some kind of ROI analysis.
5.6 Automated Test Generation

In this chapter we describe those new key areas that are relevant for automated test generation.

5.6.1 KA21 Modeling Approach

A new key area was added for the process of constructing the test model to be used in test runs. The modeling process is an essential part of an automated test generation approach, as this automatic generation can only be carried out with adequate knowledge of the test basis, which is described with the model. There are several ways to create the test model, but some approaches are considered better than others.

5.6.1.A (A) Monolithic test model

Checkpoints

5. The test model is a single, unstructured, and monolithic entity.

5.6.1.B (B) Abstract and domain-specific test model

Checkpoints

7. The test model is comprised of separate model components.

8. The modeling approach produces models in a structured fashion.

9. Problem domain has been identified.

10. The modeling approach produces models expressly for the target domain.

5.6.1.C (C) (Re-)Use of design models

Checkpoints

1. The test model can be, at least partially, derived from design models that are of a higher level of abstraction than the test model itself.

2. Design models used in the design phase, if such exist, are reused in the test modeling phase so as to avoid doing the same work again.

5.6.1.D (D) Test modeling integrated to design modeling

Checkpoints

1. Test modeling is aligned with design modeling in order to produce design models with a high level of testability.

2. Design models created in the design phase are reused in the testing phase so as to avoid doing the same work again.
5.6.2 KA22 Use of Models

As the modeling process and the quality of created test models are important when practicing automated test generation, equally important is the degree to which these models can be leveraged. Some test model might only be capable of input generation for the test object, whereas other, more advanced models might in addition verify the received output against some expected outcome. Of course, this division may not always be meaningful for all systems, but generally the capabilities of the test model factor into the extent to which this ATG paradigm can provide added value for its users.

5.6.2.A (A) Input generation with context knowledge

Testing is based on implicit (or seldom explicit) model, which understand the context of the test object. For example, in GUI testing the automation system understands that there are buttons and text fields and the automation system understand that a button can be pressed and text can be inserted and on. This level is closely related to dumb monkey testing.

**Checkpoints**

1. The test model is implicit.
2. The test model is used to generate random input values and interactions for the test object without accommodating the target domain.

5.6.2.B (B) Input generation with domain knowledge

The test model has profound knowledge of the test object. This is related to smart monkey testing.

**Checkpoints**

1. The test model is explicitly created.
2. The test model is used to generate input values and interactions expressly for the target domain.

5.6.2.C (C) Test object output verification

At this level the test model can provide information about inputs and corresponding outputs in addition to interaction invocations relevant to the domain of the test object. This can be called as a Oracle Model.

**Checkpoints**
1. The test model provides system input values and interactions with output value verifications.

5.6.3 KA23 Test Confidence

An important measure of the utility of an ATG technology is the attainable level of confidence in the quality of the test object. The ATG technology should at least be capable of covering the most critical functionality of the test object, but naturally, being able to cover basic functionality, alternative and erroneous functionalities and nonfunctional requirements of the test object is desirable.

5.6.3.A  (A) Critical (path) functionality confidence

An automated test generation system must at least test the critical behavior (smoke, basic acceptance test etc.).

Checkpoints

1. Minimum confidence is gained by testing the most critical functions of the test basis.

5.6.3.B  (B) Shallow functional confidence (Functional requirements covered)

Testing should cover all basic ways to implement required functionality.

Checkpoints

1. Every functional requirement in the test basis is covered in testing
2. The order of testing the functional requirements is affected by the importance of these requirements
3. Every functional requirement is tested at least once

5.6.3.C  (C) Thorough functional confidence

Testing should cover all basic, alternate, and erroneous ways of implementing required functionality.

Checkpoints

1. System complexity is taken into account in testing
2. Every functional requirement is tested in multiple ways, if possible
3. Every functional requirement is tested more than once
5.6.3.D (D) Non-functional requirements covered

Testing should also cover non-functional requirements and longevity aspects.

**Checkpoints**

1. Every non-functional requirement defined by the test basis is covered in testing

5.6.4 KA24 Technological and Methodological Knowledge

In order to recoup the initial investment in introducing and developing the ATG technology and secure the consistent use of the technology in the long run, the knowledge and benefits of ATG must be effectively communicated within the organization. In practical terms, the consistency of ATG usage can be fostered with appropriate training and organizational commitment.

5.6.4.A (A) Project-specific knowledge

**Checkpoints**

1. Adequate knowledge of chosen technologies and practices is present at project level

5.6.4.B (B) Adequate knowledge and training

**Checkpoints**

1. The testing personnel of an organization has been trained in the technologies and practices that are well tried, effective and suitable

5.6.4.C (C) Organizational commitment

**Checkpoints**

1. The entire organization is committed to technologies and practices that are well tried, effective and suitable

2. Positive experiences with used technologies and practices are communicated within the organization
3. Management is committed to create and sustain a positive atmosphere for the use of good practices and technologies
6 Baseline Maturity Profile for Automated Test Generation

Even though a test organization can be relatively mature, it can still fail to adapt an Automated Test Generation technique to its daily practices. One reason for failure can be, for example, the lack of commitment or training that is needed after successful pilot projects.

Table 4 presents a baseline TPI maturity profile for the successful application of ATG. The maturity profile is an estimated minimum to show what kinds of issues are important for a testing organization that is moving toward automatic test generation.

Table 4: Baseline maturity profile for Automated Test Generation

<table>
<thead>
<tr>
<th>Key Area</th>
<th>Scale</th>
<th>0 (Initial)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Strategy</td>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Life Cycle Model</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moment of Involvement</td>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estimating and Planning</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Specification Techniques</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static Test Techniques</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metrics</td>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Automation</td>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Environment</td>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Office Environment</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commitment and Motivation</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Functions and Training</td>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scope of Methodology</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reporting</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defect Management</td>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testware Management</td>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Process Management</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluation</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low-level Testing</td>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Since ATG is one way to use test automation, a strong emphasis is laid on Key Area 8 “Test Automation”. Even though Table 4 shows that the corresponding maturity level should be Managed (B), it would be even better if it were Optimizing (C).
7 Future of Test Process Improvement - TPI NEXT

Even though TPI has been very successful and widely used in industry, there is still plenty of room for improvement. As TPI has been applied by many users, some overlapping and less important areas have been identified. In addition, the original had been proven to be a little bit rigid and its adaptability to different business models and needs was inadequate. The new version of TPI [2], TPI NEXT [5][6], has only 16 key areas and 157 checkpoint questions, as opposed to the 20 key areas and almost 300 checkpoints of the original TPI.

7.1.1 Something old

In TPI NEXT the basic structure was kept unchanged, but some key areas were combined because they were closely related to each other. As ‘Lice-Cycle Model’ proved to be closely related to ‘Test Process management’, they were combined into new ‘Test Process Management’. Since ‘Commitment and Motivation’ measured both testers' and stakeholders', i.e. two totally distinct groups', motivation and commitment, the key area was split into two new key areas: ‘Stakeholder commitment’ and ‘Tester Professionalism’. These new key areas describe better what the old key area tried to do.

The original TPI also did not properly address the position and role of a test organization inside the entire organization, regardless of the importance of this issue. This is now better accommodated with the creation of a new key area called ‘Test Organization’. It covers organizational issues better since those issues are now gathered in one place.

Because the process improvement can be focused on one particular activity, like evaluation, it is not practical to have a focus area or scope as a key area in TPI. Thus the key areas of ‘Low-level testing’ and ‘Evaluation’ have been removed from the TPI NEXT, which was a good move since they did not seem to fit in. However, this way of seeing process improvement might have some ramifications, which can influence the ATG adaptation to the TPI NEXT.

7.2 Other updates

Maturity levels were redefined and now there are 4 levels of maturity:

0. Initial; “Ad hoc activities”
1. Controlled; “Doing the right things”
2. Efficient; “Doing things the right way”
3. Optimizing; “Continuously adapting to ever-changing circumstances”

These levels visualize better the overall test process maturity than the original TPI did.
Table 5: TPI NEXT maturity matrix

7.3 Something new

There are two important additions in TPI NEXT; enablers and clusters. Enablers are checkpoints that show where the improvement of the test process and that of the overall development process can have mutual benefits. The purpose of enablers is to enhance and speed up the improvement process and they can be indicators for the development process in terms of how to gain some additional value if the improvement effort is focused correctly.

A cluster is a method that enhances the idea of stepwise process improvement. A cluster is a group of checkpoints that have a mutual relationship and serve as a single improvement step. For example, there is a cluster group for CMMI [7], which gives steps to improve the testing process in synchrony with CMMI levels 1 through 5. An organization can create cluster groups for various businesses needs that serve organizations’ improvement effort the best. However, when creating clusters, the creator must be aware that many checkpoints have relationships with others checkpoints and this creates a restriction on what can be placed in different clusters.
8 Glossary

This glossary contains terms and definitions that are available in TMap Glossary (http://eng.tmap.net/Home/TMap/Glossary.jsp). Some terms, which are not common in general testing literature are also included in the list.

ATG basis
The ATG basis is the information that covers the test basis and conventional test information (test specs, scripts etc.)

Infrastructure (suitability of)
The suitability of hardware, network, systems software and DBMS for the application concerned and the degree to which the elements of this infrastructure interrelate.

Intake of test object
This is generally known as test input for the test object (test data and commands?).

Test basis
The test basis is the information that defines the required system behavior.

Test case
Used to examine whether the system displays the desired behavior under specific circumstances.

Test design technique
A standardized method of deriving test cases from a particular test basis that will achieve a certain coverage.

Test environment
A composition of parts, such as hardware and software, connections, environment data, maintenance tools and management processes in which a test is carried out.

Test goal
A goal that, for the client, is relevant for testing, often formulated in terms of business processes supported by IT, realized user requirements or use cases, critical success factors, change proposals or cited risks to be covered.

Test infrastructure
Consists of the facilities and resources necessary to facilitate the satisfactory execution of the test. A distinction is made between test environments, test tools and workplaces.

Test level
A test level is a group of test activities that are managed and executed collectively.

Test line
The operational organization to provide test services to one or more clients. A test line has a fixed team of testers, infrastructure, test tools and standardized work procedures.

Test model
The test model is a computational abstraction of the behavior of the test object and also the data that is required during the execution of the test object.
**Test object**
The test object is the information system (or part thereof) to be tested.

**Test organization**
The whole of the test functions, facilities, procedures and activities including their relationships.
9 Bibliography


