Integrating Software Testing Standard ISO/IEC/IEEE 29119 to Agile Development

Ning Chen¹, Ethan W. Chen¹ and Ian S. Chen²
¹Department of Computer Science, California State University, Fullerton, California, USA
²Raytheon Company, Tucson, Arizona, USA

Abstract - The IEEE standard 29119 on Software and Systems Engineering - Software Testing which replaces an older standard of IEEE Std 829 and others is designed with the need of agile process in mind. It provides an explanation on Agile projects and some suggestions on integrating the standard to Agile process. Nevertheless, integrating the standard to Agile, still is not that straightforward and may need further elaboration. This paper addresses the following issues: the needs for a testing standard, the mechanism that integrates the testing standard to Agile, and on how to tailor the conformance to a proper level that involves the maturity levels of the Test Maturity Model Integration (TMMi). The paper concludes with a suggested tailored conformance plan that is suitable for a typical Agile Development.

Keyword: IEEE Std 29119, Software Testing, conformance, Agile, TMMi

1 Introduction

Testing is an integral part of the software development process. ISO/IEC/IEEE 29119 is a relatively new standard for software testing with the most recent part published in 2016 [1]. As a new international standard, IEEE 29119 becomes the benchmark upon which testing processes are measured against. Organizations are then influenced to adhere to the standard to some degree and implement IEEE 29119 conformance within their existing software development process.

Agile methodology is one of the most popular software development philosophies and widely used. Per a recent 2017 survey, 94% of respondents claimed that their organizations practiced Agile [2]. As a result, many companies will face or are currently facing the task of implementing IEEE 29119 within their Agile process. One of the most notable standards that IEEE 29119 is replacing is IEEE 829-2008, which describes test documentation standards [3]. The old standard, IEEE 829, is deeply rooted in the traditional Waterfall development lifecycle, but is not entirely at odds with implementation into Agile [4]. Though IEEE 29119 was written with more consideration for Agile in comparison to IEEE 829, it does not clearly describe how it can be implemented into Agile. The question then arises of whether IEEE 29119 can be easily integrated into Agile and how.

Though IEEE 29119 does not necessarily conflict with Agile, it is possible that full conformance to the standard, as defined in IEEE 29119, may be too cumbersome for proper Agile methodology. This produces a conundrum where extensive adherence to testing standards is required to be fully compliant with IEEE 29119, but in doing so would violate Agile principles. At the same time, it would not be sensible to completely forgo any testing standards in Agile. Some industry surveys estimate that testing comprises up to 30 to 50 percent of the total cost of development [5]. As a result, some testing standard is desired in order to properly exhibit the results of and to justify the expense invested in testing.

Though IEEE 29119 provides some leeway in this problem with the option of tailored conformance, the implementation of tailored conformance is arguably too loose. Tailored conformance in IEEE 29119 is defined to be completely dependent on a specific organization and/or project needs, and thus loses its utility as a standard. Since Agile holds significant market-share in the community, a standard adoption of IEEE 29119 within Agile is needed to maintain parity between different organizations. If no Agile standard of IEEE 29119 is available, implementations of IEEE 29119 tailored conformance across multiple organizations may be too radically different to be properly compared.

This paper starts with an analysis of IEEE 29119 and Agile methodology. We then explore the need for a testing standard, propose an integration of IEEE 29119 within Agile, and describe IEEE 29119 conformance maturity levels related to the Test Maturity Model Integration (TMMi).

2 What is IEEE Std 29119?

IEEE 29119 is an internationally agreed set of standards with the purpose of supporting software testing. IEEE 29119 consists of five different parts:

1. IEEE 29119-1: Concepts & Definitions
2. IEEE 29119-2: Test Processes
3. IEEE 29119-3: Test Documentation
4. IEEE 29119-4: Test Techniques
5. IEEE 29119-5: Keyword Driven Testing

IEEE 29119 is intended to replace the following existing standards for software testing:
IEEE 29119: Concepts & Definitions – This document correlates to BS 7925-1 and serves as a glossary of definitions of testing terms and concepts used in the overall IEEE 29119 standard.

IEEE 29119-2: Test Processes – This document describes software testing processes at multiple levels. The processes are meant to be generic as to be able to be implemented in any organization for any kind of software testing and for any type of software development life cycle model. Most notably, IEEE 29119-2 focuses on a risk-based approach to testing with the primary goal of risk mitigation. The standard test process model described is multi-layered with a top level organizational test process, a middle level test management process, and a bottom level of dynamic test processes [Figure 1].

IEEE 29119-3: Test Documentation – This part of IEEE 29119 provides templates and examples of test documentation. These work products are aligned with the test processes described in IEEE 29119-2 and cover the organizational level, test management level, and dynamic test level. Examples and suggestions are provided for both Agile and traditional software development life cycle models.

IEEE 29119-4: Test Techniques – A description of software test design techniques that align with the test processes described in IEEE 29119-2 is given. The document describes how to derive test conditions, test coverage items, and test cases.

IEEE 29119-5: Keyword Driven Testing – The most recently published portion of IEEE 29119 describes the usage of keyword-driven testing as a strategy of modular testing for usage of test automation frameworks.

IEEE Std 29119 lays out a set of requirements to be considered as adherent to the standard. Though IEEE 29119 provides examples of how test documentation might look like in Agile or traditional software development lifecycle models (found in IEEE 29119-3), it does not provide specific instruction on how IEEE 29119 should be implemented. Regarding test process, IEEE 29119-2 sets standards of what forms of test processes should be defined at multiple levels [Figure 1] but no specifics on integration into various software development lifecycle models. Similarly, IEEE 29119-3 describes what work products in the form on documentation should be produced throughout the testing process, but provides minimal guidance on how it should be implemented. IEEE 29119-4 describes various test techniques that can be used, but does not specify any particular requirements as to which are universally needed. Rather the user is left to pick and choose which techniques are appropriate and suitable for their situation. Lastly, IEEE 29119-5 defines a modular approach to describing test cases through the Keyword-Driven Testing framework which lends itself to test automation and is a specific implementation of the test process model described in IEEE 2119-2. Similarly, to the rest of IEEE 29119, the testing framework itself is described in detail, but the specific integration of the framework into different types of software development lifecycle models is not.

3 What is Agile Development?

Agile development is a broad term encompassing a set of practices that promote the values espoused in the Agile Manifesto [6]. From the Agile Manifesto [Figure 2] are derived the Twelve Principles of Agile that make up the Agile philosophy. In total, Agile is meant to enhance the ability to effectively respond to the constant change present in the software development process. Agile advocates continuous, small incremental releases of working software. Most importantly Agile describes the methods, or the “how”, in software development (e.g. Scrum, Kanban).

3.1 Software Testing in Agile Development

In Agile Development, software testing is performed differently in comparison to traditional testing methodology like Waterfall Model. Instead of testing performed as a separate and distinct phase after coding, Agile software testing is performed in concurrence with coding. Furthermore, testing is integrated into the development team as well as relevant stakeholders and the distinction between developers and testers is blurred.

In Agile methods, there is no true definitive or standard testing strategy. Though Test Driven Development (TDD) is
an extremely popular method of software testing in Agile, it is not required, strictly speaking, to adhere to the Agile Manifesto. The first line of the Agile Manifesto describes this succinctly: "Individuals and interactions over processes and tools". Agile does not technically prescribe any specific testing processes only that the software development process remains responsive to change. Literature has expanded on this topic and produced recommendations and trends on best practices of implementing testing into Agile [7] [8].

3.2 The Role and Need of Testing Standard in Agile Development

Agile, in a strict interpretation, does not prescribe or require any sort of testing standard. This is not to say that a testing standard would not be valuable in an Agile environment. A standard provides common ground upon where the community can agree upon what practices are necessary to say that the overall testing process is sound.

Individuals and interactions over processes and tools - Though the Agile Manifesto does prefer individuals and interactions over processes, that does not mean that testing processes are not needed. In the end, having standardized testing processes and organization provides structure to the overall testing effort. Utilizing standard processes also does not run contrary to Agile as long as it remains flexible and does not become rigid to the point that responsiveness to change is affected, thus violating Agile philosophy.

Working software over comprehensive documentation - Though Agile does explicitly value "Working software over comprehensive documentation", the total absence of any testing documentation is not desirable. Some documentation is desired to have records that can demonstrate testing was adequately performed. At the same time, excessive documentation would run contrary to Agile. Agile organizations then must individually determine what level of documentation is needed to satisfy their own internal documentation requirements while remaining Agile. With such a broad specification, a testing standard would be useful here.

Customer collaboration over contract negotiation - Though this particular part of the Agile Manifesto does not explicitly involve testing, customer collaboration can be an important part of the Agile testing process. For example, Acceptance Test Driven Development, where customer engagement is key to testing, can be implemented into Agile though it is not necessarily a standard.

Responding to change over following a plan - Similarly, while more value is placed on the capability to respond to change, this does not mean that a plan has no value. A test plan does provide value in an Agile environment as long as it does not hinder responsiveness to change.

Agile philosophy describes how testing should be implemented. It should be continuous and integrated into the development lifecycle. What Agile does not specify is what processes should be carried out or what work-products should be produced throughout the testing process. In that sense, testing standards do not conflict with Agile development and instead can be complementary to Agile. The question then becomes how should testing standards, in particular IEEE 29119, be implemented in Agile environments.

4 How to Integrate?

Testing in Agile development aims at different goals than that of traditional iteration life process models (e.g., Water Fall) [7]. There are no firm requirements at the beginning and typically no solid, fixed architecture plan either. Furthermore, one of the important goals in each iteration (e.g., Sprint) is to have a “working software”. The term “working software” implies the existing of requirements and the process of verifying those requirements even they are very informal, not in the form of formal documentation. This opens the idea that if the development activity can be iterative and incremental, then the forming of test plans, test documentation, and test tool policy can also be iterative and incremental [4] [9]. We present an effort in forming of plans, documentation, and policy in a Scrum setting as shown below.

4.1 Sprints in Scrum

We briefly describe the Scrum process first. The Scrum development is done in time-boxed efforts called Scrum sprints. As the business requirements evolve incrementally, the Product Backlog (e.g., user stories) forms. Each sprint aims for reducing some project backlog in one to four weeks of time. Although the business requirements typically focus on feature and functionality of the software product, it is possible to treat the need to have processes, documentation, techniques and automation related keyword driven testing framework as part of the product backlog and allocate special sprints, for example, sprint zero or other sprints, for this purpose.

4.2 Processes, Documentation, Techniques and Keyword Driven Framework

Part 1 of 29119 describes the concepts and definitions and is informative. Part 2 to Part 5 involve the conformance clauses. Part 2 is related to test processes and has three major sections: Organization Test Process, Test Management Processes, and Dynamic Test Processes. An easy interpretation of those processes is that we have some steps of doing things (i.e., activities) for some purpose. If we record the way of doing things and their outcomes, documentation typically is a natural artifact. Documentation logically leads to the Part 3 of 29119. Part 3 requires three sub-sections: Organizational Test Process Documentation, Test Management Processes Documentation, and Dynamic Test Processes Documentation. Part 4 of 29119 describes the test techniques (e.g., how one executes the tests). Part 5 of 29119 is titled Keyword Driven Testing. It is designed to pave the ground work for test automation. Maybe naming it “Test Automation Framework Based on Keywords” is more intuitive and makes it more Agile friendly since most Agile projects, if
not all, involve test automation. We decide to list relevant section names of Parts 2 to 5 below just to show the first impression that most people get – quite massive and almost for certain anti-Agile!

**Processes (29119 Part 2)**

**Organizational Test Process**

Develop Organizational Test Specification (OT1)
Monitor and Control Use of Organizational Test Specification (OT2)
Update Organizational Test Specification (OT3)

**Test Management Processes**

**Test planning**
- Understand Context (TP1)
- Organize Test Plan Development (TP2)
- Identify and Analyze Risks (TP3)
- Identify Risk Mitigation Approaches (TP4)
- Design Test Strategy (TP5)
- Determine Staffing and Scheduling (TP6)
- Record Test Plan (TP7)
- Gain Consensus on Test Plan (TP8)
- Communicate Test Plan and Make Available (TP9)

**Test monitoring and control**
- Set-Up (TMC1)
- Monitor (TMC2)
- Control (TMC3)
- Report (TMC4)

**Test completion**
- Archive Test Assets (TC1)
- Clean Up Test Environment (TC2)
- Identify Lessons Learned (TC3)
- Report Test Completion (TC4)

**Dynamic Test Processes**

**Test design & implementation**
- Identify Feature Sets (TD1)
- Derive Test Conditions (TD2)
- Derive Test Coverage Items (TD3)
- Derive Test Cases (TD4)
- Assemble Test Sets (TD5)
- Derive Test Procedures (TD6)

**Test environment set-up & maintenance**
- Establish Test Environment (ES1)
- Maintain Test Environment (ES2)

**Test execution**
- Execute Test Procedure(s) (TE1)
- Compare Test Results (TE2)
- Record Test Execution (TE3)

**Test incident reporting**
- Analyze Test Result(s) (IR1)
- Create/Update Incident Report (IR2)

**Documentation (29119 Part 3)**

**Organizational Test Process Documentation**

**Test Policy**
- Document specific information
- Overview
- Unique identification of document

**Test Management Processes Documentation**

**Test Plan**
- Overview
- Document specific information (unique id, issuing organization…)
- Introduction (scope, references, glossary…)
- Context of the testing (projects/test sub-processes, test items, test scope, assumptions and constraints…)
- Testing communication
- Risk register (product risks, project risks, …)
- Test strategy (test sub-processes, test deliverables, …)
- Testing activities and estimates
Staffing (roles, activities, and responsibilities, hiring needs…)

Test Status Report
   Overview
   Document specific information (unique id, issuing organization…)
   Scope
   Test status (reporting period, progress against test plan, …)

Test Completion Report.
   Overview
   Document specific information (unique id, issuing organization…)
   Introduction (scope, references…)
   Test performed (summary, deviations…)

Dynamic Test Processes Documentation
   Test design specification
   Test case specification
   Test procedure specification
   Test data requirements
   Test environment requirements
   Test data readiness report
   Test environment readiness report
   Actual results
   Test results
   Test execution log
   Test incident reporting

Test Techniques (29119 Part 4)
   “An organization could choose to conform to one technique, such as boundary value analysis. In this scenario, the organization would only be required to provide evidence that they have met the requirements of that one technique…”
   Identify Feature Sets (TD1)
   Derive Test Conditions (TD2)
   Derive Test Coverage Items (TD3)
   Derive Test Cases (TD4)

Keyword Driven Testing Framework (29119 Part 5)
   Basic attributes
   General attributes
   (a) There shall be documentation recorded describing each keyword.
   (b) There shall be documentation recorded for the parameters of each keyword.

   Dedicated keyword-driven editor (tool)
   (a) Within the keyword-driven editor, non-composite keywords shall be displayed with their associated actions.
   (b) For keywords which have been defined with lower level keywords, the user shall be able to access this definition within the keyword-driven editor.

   Decomposer and data sequencer
   (a) The decomposer shall be able to process parameters, including assuring that the parameters associated with the higher-level keywords are decomposed and associated with the lower-level keywords.

Manual test assistant (tool)
   (a) The manual test assistant shall support manual test execution based on the defined test cases.

   Tool bridge
   (a) The tool bridge shall provide the test execution engine with the appropriate execution code to execute the test cases.

   Test execution engine
   (a) Keywords that do not express conditions or loops within a test case shall be executed sequentially starting with the first keyword.

   Keyword library
   (a) The keyword library shall support the definition of keywords that includes the basic attributes of name, description and parameters.

   Script repository
   (a) The script repository shall support the storage of keyword execution code.

4.3 Is It Really Anti-Agile?
   The first impression sometimes may not be always correct. Is Std 29119 really Anti-Agile? Let’s take a look at the Processes part (29119 Part 2). Part 2 has three sub-parts (Organizational Test Process, Test Management Processes and Dynamic Test Processes). The sub-part of Test Management Processes is divided further into Test planning, Test monitoring and control, and Test completion. The sub-part of Dynamic Test Processes is divided further into Test design & implementation, Test environment set-up & maintenance, Test execution, and Test incident reporting. At this point a novice user of 29119 is probably overwhelmed by a plethora of similar terms. For convenience, Part 2 of 29119 provides abbreviations for all required sections as:
   OT1 to OT3, TP1 to TP9, TMC1 to TMC4, TD1 to TD6, ES1 to ES2, TE1 to TE3 and IR1 to IR2.

   To make the “process” even more heavy, not lighter, one may soon realize that the Dynamic Test Processes need to be repeated multiple times, for example, each time for unit, integration, system and acceptance test. This means at least four times of (TD1 to TD6, ES1 to ES2, TE1 to TE3 and IR1 to IR2).

   We think the abbreviations actually provide a hint that points to a possible light-weight usage of the standard – we can view them as a “pre-flight checklist.” It may take a competent pilot having years of training to be familiar with the pre-flight checklist. Nevertheless, the actual checking of the pre-flight checklist by a competent pilot only takes a few minutes. We can also use the Agile Development itself as an analogy. Although it takes four years of training to train a competent software developer for most of the professionals who hold B.S. degree in Computer Science, it only takes a short sprint for those professionals to produce a small but working
software product. By the same logic, it is hard not to accept that a well-trained tester can quickly finish the “pre-flight checklist” of the test processes. We envision that Part 2 of 29119 can be realized in a form of couple pages of tables in Sprint Zero. Its contents can be incrementally filled and improved in the subsequent sprints.

4.4 How About Part 3 of 29119 (Documentation)?

Part 3 of 29119 requires the following three sub-parts: Organizational Test Process Documentation, Test Management Processes Documentation, and Dynamic Test Processes Documentation. How do we integrate Part 3 to Agile Development?

Before we answer that question, it is probably apparent that those three sub-parts of documentation match exactly to the three sub-parts of Part 2 of 29119 (Processes). This observation may provide a hint on how we integrate Part 3 of 29119 to Agile Development. Part 3 (Documentation) is just a natural “result” of Part 2 (Processes). For example, when one executes the “pre-flight checklist” of TP1 to TP9 (Test Planning under Test Management Processes), a natural result will be Test Plan (one sub-part of Test Management Processes Documentation of Part 3 of 29119). We believe that it is highly feasible to have a second set of tables (just a couple of pages) that match the “pre-flight checklist/process” for briefly recording the results of the activities of the “pre-flight checking” (processes).

4.5 How About Parts 4 (Test Technique) and 5 (Keyword Driven Testing) of 29119?

Parts 4 and 5 of 29119 cover Test Technique and Keyword Driven Testing. For Agile Development it is relatively easy to conform the requirements of Parts 4 and 5. The reason is that any Agile Development project involves at least one typical test technique (for example, Boundary Value Testing) and this almost automatically fulfills the requirements of Part 4. Furthermore, most of the Agile projects, if not all, employs the use of automated test tools (proprietary or open source) that most likely fit into the Keyword-Driven Testing Framework as outlined in Part 5 of 29119.

5 Conformance

Even readers who accept our argument that the heavy processes and documentation can be “not-that-heavy-at-all”, a lingering question remains – Is there a proper level of tailored conformance, since full conformance may not be desirable? Most professionals may respond that it depends! We envision that another line of research Test Maturity Model Integration (TMMi) [10] may play a role in answering this question.

5.1 What is TMMi?

TMMi [10], same as its older sister CMMi (Capacity Maturity Model Integration), has five levels of maturity: Initial, Managed, Defined, Measured, and Optimized as depicted in Fig. 3.

5.2 Why Add TMMi in Integrating 29119 to Agile Development?

At its core value TMMi is not a standard and it does not impose rules, nor it demands “proof of compliance.” TMMi’s main goal is to encourage “changes in testing” that gives value to the business objectives [11]. IEEE 29119 indeed is a standard and it does demand proof of compliance. Obviously, there are no fundamental conflicts between TMMi and 29119. Furthermore, TMMi does not conflict with Agile Development, simply because, again, the former, same as 29119, focuses on the what problem and the latter mainly aims at the how issue. One may still question the need to bring TMMi into the picture. What is the main value that TMMi brings to the 29119 and Agile Development integration? We think the staged maturity model gives discrete, clear, and reachable targets that encourage changes in testing to increase values in business. The discrete, clear, and reachable targets also encourage Agile Development to seek conformance of Std 29119 in tailored, incremental steps which fit naturally into the Agile mentality.

6 Integration Example

In this section we propose an integration example (tailored conformance) of 29119 and Agile Development that targets the TMMi Level 2 maturity model (Managed). TMMi Level 2 Managed targets five process areas (PAs) and their corresponding goals (called Specific Goals, SP) as below:

Test Police and Strategy (Establish a Test Policy, establish a Test Strategy, and Establish Test Performance Indicators)

Test Planning (Perform Risk Assessment, Establish a Test Approach, Establish Test Estimates, Develop a Test Plan, Obtain Commitment to the Test Plan)

Test Monitoring and Control (Monitor Test Progress Against Plan, Monitor Product Quality Against Plan and Expectations, and Manage Corrective Actions to Closure)

Test Design and Execution (Perform Test Analysis and Design using Test Design Techniques, Perform Test Implementation, Perform Test Execution, and Manage Test Incidents to Closure)

Test Environment (Develop Test Environment Requirements, Perform Test Environment Implementation, and Manage and Control Test Environments)
6.1 Tailored Conformance at TMMi Level 2 (Managed)

We map the TMMi Level 2 (Managed) process areas (PAs) and their specific goals (SPs) to Part 2 of 29119 in the table below:

<table>
<thead>
<tr>
<th>TMMi Level2 (Managed) Process Area (PA)</th>
<th>Specific Goal (SP)</th>
<th>Part 2 of 29119 (Processes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Police and Strategy</td>
<td>Establish a Test Policy</td>
<td>TP2</td>
</tr>
<tr>
<td></td>
<td>Establish a Test Strategy</td>
<td>TP5</td>
</tr>
<tr>
<td></td>
<td>Establish Test Performance Indicators</td>
<td>TP5</td>
</tr>
<tr>
<td>Test Planning</td>
<td>Perform Risk Assessment</td>
<td>TP3, TP4, TMC1</td>
</tr>
<tr>
<td></td>
<td>Establish a Test Approach</td>
<td>TMC1</td>
</tr>
<tr>
<td></td>
<td>Establish Test Estimates</td>
<td>TP6</td>
</tr>
<tr>
<td></td>
<td>Develop a Test Plan</td>
<td>TP2, TP7</td>
</tr>
<tr>
<td></td>
<td>Obtain Commitment to the Test Plan</td>
<td>TP8, TP9</td>
</tr>
<tr>
<td>Test Monitoring and Control</td>
<td>Monitor Test Progress Against Plan</td>
<td>TMC2, TMC4</td>
</tr>
<tr>
<td></td>
<td>Monitor Product Quality Against Plan and Expectations</td>
<td>TMC1, TMC2, TMC4</td>
</tr>
<tr>
<td></td>
<td>Manage Corrective Actions to Closure</td>
<td>TMC3</td>
</tr>
<tr>
<td>Test Design and Execution</td>
<td>Perform Test Analysis and Design using Test Design Techniques</td>
<td>TD1,2, 3</td>
</tr>
<tr>
<td></td>
<td>Perform Test Implementation</td>
<td>TD4, 5</td>
</tr>
<tr>
<td></td>
<td>Perform Test Execution, and Manage Test Incidents to Closure</td>
<td>TD6</td>
</tr>
<tr>
<td>Test Environment</td>
<td>Develop Test Environment Requirements</td>
<td>ES1</td>
</tr>
<tr>
<td></td>
<td>Perform Test Environment Implementation, and Manage Control Test Environments</td>
<td>ES2</td>
</tr>
</tbody>
</table>

Part 3 of 29119 can be fulfilled by the above corresponding processes. Part 4 and Part 5 of 29119, as discussed before, almost always are met by a typical Agile Development. We envision that this tailored conformance example may serve as a test template for most typical Agile projects.

7 Conclusion

The IEEE standard 29119 on Software and Systems Engineering - Software Testing has five major parts: Concepts and Definitions, Test Processes, Test Documentation, Test Techniques and Keyword-Driven Testing. It was designed to suit the needs of different life cycle models, for example, Agile, Evolutionary and Sequential (i.e., the waterfall model). The life cycle model can be viewed as an answer to the how and when questions (e.g., how and when to perform some desirable activities). The 29119 standard can be viewed as an answer to the what questions (e.g., what are the desirable activities needed to be done so that it is not too far away from the norm of the community).

For sure, it is very reasonable to argue that a standard may not be perfect, especially when the standard urges, if not dictates, everyone in the community to meet the requirement on the question of what needs to be done. The answer to the question on what needs to be done always is debatable and any answer will have room for improvement. On the other hand, it is hard to argue that there is no need to have a standard in a community in which a common platform is needed so that stakeholders have a common ground in conducting, for example, discussion, brainstorming, and evaluation.

Our premise is that in Agile, although we value other characteristics more, we still need some processes, plans, documentation, and tools (i.e., it really cannot be zero, nothing). If our premise is valid, the next logical question then is "What is the proper amount of processes, tools, documentation, and plans suitable for Agile development?" At this point, one may suspect that there are no answers for that—it highly depends on the agile project at hand, and/or the organization, the client, the budget, etc.

In this research we propose to bring another line of research, TMMi (Test Maturity Model Integration), into the picture. TMMi recommends five maturity levels: Initial, Managed, Defined, Measured and Optimized. The proper amount of processes, plans, documentation and tools in an Agile setting can be reasonably gauged by the maturity level as recommended by TMMi. The paper concludes with an example of tailored conformance of IEEE 29119 that targets TMMi maturity level 2 (i.e., Managed) and likely is suitable for most of the Agile projects.

8 References