An Industrial Survey on Contemporary Aspects of Software Testing

Adnan Causevic, Daniel Sundmark, Sasikumar Punnekkat
Mälardalen University, School of Innovation, Design and Engineering,
Västerås, Sweden
{adnan.causevic, daniel.sundmark, sasikumar.punnekkat}@mdh.se

Abstract — **Software testing is a major source of expense in software projects and a proper testing process is a critical ingredient in the cost-efficient development of high-quality software. Contemporary aspects, such as the introduction of a more lightweight process, trends towards distributed development, and the rapid increase of software in embedded and safety-critical systems, challenge the testing process in unexpected manners. To our knowledge, there are very few studies focusing on these aspects in relation to testing as perceived by different contributors in the software development process.**

This paper qualitatively and quantitatively analyses data from an industrial questionnaire survey, with a focus on current practices and preferences on contemporary aspects of software testing. Specifically, the analysis focuses on perceptions of the software testing process in different categories of respondents. Categorization of respondents is based on safety-criticality, agility, distribution of development, and application domain. While confirming some of the commonly acknowledged facts, our findings also reveal notable discrepancies between preferred and actual testing practices. We believe continued research efforts are essential to provide guidelines in the adaptation of the testing process to take care of these discrepancies, thus improving the quality and efficiency of the software development.

Keywords-Software testing, testing practices, agile testing.

I. INTRODUCTION

Software testing, as a practice, has been able to successfully evolve over time and provide efficient and constant support for improvements in software quality. On the other hand, testing is still notorious for its massive resource consumption within software projects. To this date, much of the research efforts on software testing have been focusing on designing new techniques, as well as investigating their effectiveness in real development contexts. However, during the entire history of software development, testing methods and techniques have struggled to keep up with the ever faster evolution and trends in software development paradigms. We cannot expect any favourable change in this state of affairs, unless a conscious effort is made in anticipating the trends, learning the stakeholder mindsets, and pinpointing the problem areas. It is our belief that such an effort could help in efficiently allocating the testing resources toward a specific context or in proactively deciding the testing research agenda in general. To our knowledge, there exist no detailed investigations with such a perspective. The research presented in this paper is a small step in this direction, in that it specially focuses on the stakeholder perspectives on some contemporary aspects related to testing. The specific research question we address in this paper is: *Is it possible to identify and list main discrepancies between current and preferred testing practices that could be considered as obstacles for software testing practitioners?*

By qualitatively and quantitatively analysing the results of a recent questionnaire survey on practices and preferences in industrial software development, with respect to the above research question, we have identified a number of areas and practices where the preferred practice significantly differs from what is perceived as the actual current practice. We believe that these areas and practices assist in pointing out directions for future research within software testing.

The contributions of this paper are three-fold:

- A qualitative analysis on practices and preferences in testing of different contemporary categories of software development professionals (Section III).
- A qualitative analysis on techniques and tools used in contemporary testing (Section IV).
- A quantitative analysis of satisfaction with current testing practice among different categories of respondents (Section V).

II. RESEARCH METHOD

We base our analyses in this paper on data from a recent industrial survey on software development practices and preferences. This survey was a combined effort of several researchers with diverse research foci, all integrated to one study to minimise the responders time. The survey was performed using a web-based questionnaire, and the invitation was distributed among industrial software development companies using, e.g., the FLEXI[1] and NESSI[2] European project networks. More information about the questionnaire, as well as all data, is available as a technical report [5]. The questions pertinent for our research were embedded in the larger set of questions and were formulated in such a way as to provide the list of discrepancies indirectly rather than asking the questions in a direct way which could be either provocative or could result in a no response since some of the respondents may not want to present themselves as opinionated.

[1] www.flexi-itea2.org
[2] www.nessi-europe.com

TABLE I. CATEGORIZATION OF RESPONDENTS

Categorizing Criterion	Categorizing question and response		Category
	Question	Response	
Agility of Development Process	Our current software development process is:	"Agile"	Agile
		Other	Non-agile
Distribution of Development	In our team: all of the team members are collocated in one building	Yes	Distributed
		No	Non-distributed
Safety-Criticality of Product(s)	If the software developed in our current project fails, the maximum damage could be the loss of:	"Many lives" or "A single life"	Safety-critical
		Other	Non-safety-critical
Amount of testing performed by Respondent	At work, I perform the following activities [indicate how often on a scale of 0 to 7]: [testing]	1-7	Testers
		0	Non-testers
Domain of Product(s)	The software we build in our project is:	"Web software"	Web
		"Desktop software"	Desktop
		"Embedded software"	Embedded

A. *Categorization of Respondents*

In our analysis, we categorize survey respondents according to five aspects of contemporary software development, namely:

1) Agility of Development Process
2) Distribution of Development
3) Domain of Product(s)
4) Safety-Criticality of Product(s)
5) Amount of Testing performed by Respondent

In order to categorize respondents according to these aspects, we make use of a set of categorizing questions. For example, a respondent belongs in the "Agile" category of respondents if the answer to the question "*Our current software development process is:*" is "*Agile*".

Note that any respondent may be included in several non-mutually exclusive categories. For example, a respondent might be categorized as a *tester,* working with *safety-critical* software in a *non-distributed development* (examples of mutually exclusive categories are *testers* and *non-testers,* and *distributed* and *non-distributed*). The domain categories are not mutually exclusive, as some respondent companies develop software for multiple domains. This cross-coupling between categories was an intentional outcome of our research objective of unearthing more inter-related answers from the respondents indirectly.

The categorization of respondents and the categorizing questions can be viewed in TABLE I.

B. *Question Selection*

Since the amount of questions in the survey data is quite large (a total of 260 questions were included in the questionnaire), we wanted to focus only on questions that explicitly or implicitly related to the testing process. For the explicitly test-related questions, this process was trivial, but for the more implicit questions, the process was subjective. Our rationale was that, for a question to be test-related, the practice queried on needs to directly affect the testing process. An example of such an implicitly test-related question is the question *"Management should encourage regular interaction between developers and customers/business people"*, since regular interaction between developers and customers directly may affect how acceptance testing is performed.

Moreover, for each contemporary aspect, we selected and studied a subset of test-related questions that specifically applied to that particular contemporary aspect. For example, the question *"We never have to wait for source code in order to start the testing process"* is highly interesting in a distributed development context, where communication between different development teams may be impaired by, e.g., geographical and cultural distances within the organisation.

C. *Scales Used for Answers*

Respondents were providing their opinion by selecting one of the options from a given scale of answers against each question. Two different scales (with 7 or 8 divisions) were used in our survey depending on the type of question. The 7-scale options represent: "Very strongly disagree", "Strongly disagree", "Disagree", "Neither agree nor disagree", "Agree", "Strongly agree" and "Very strongly agree". For certain analyses, 7-scale answers were mapped to numerical values in the (-3, 3) interval. This would mean that a "-2" value should be interpreted as the respondent strongly disagrees with the statement provided in the question, whereas a "1" should be interpreted as the respondent agrees. The 8-scale answers were used for questions in which respondent were asked to provide the level of usage for some testing types. This way they could choose an option from 0 to 7 where 0 means they never use it and 7 they always use it.

In Section III, TABLE III. to TABLE VII. present the data from the questions where the used scale is implicitly assumed from left to right as the headings of the columns with numeric values. The numerical values in turn represent the number of respondents in that particular group who have answered that particular question with the same answer option.

III. Testing Practices and Preferences

For a better understanding of our targeted respondents group, a general overview of respondent demographics for the complete survey is presented in TABLE II.

TABLE II. Respondent Demographics

Gender	
Male	**73**
Female	**10**
Age	
25-29	**13**
30-34	**22**
35-39	**25**
40-49	**14**
50-59	**6**
Educational qualification	
Undergraduate or lower	**3**
Bachelor degree	**21**
Postgraduate degree	**45**
PhD or above	**14**
IT work experience	
1-4 years	**16**
5-8 years	**22**
9-12 years	**18**
13-16 years	**14**
More than 16 years	**13**
Team size	
I am not in a team	**6**
1-5 people	**25**
6-10 people	**22**
11-15 people	**10**
16-20 people	**6**
21-50 people	**9**
more than 50 people	**4**
End product of project is	
a software part/component which is to be integrated	**15**
a software service	**4**
a software system that will be used by end users	**39**
other	**5**

Now we analyze data regarding practices and preferences in testing in the different categories of respondents. Please note that some questions are conditional based on earlier responses which as a result provide less number of responders compared to the complete survey. This analysis is performed on five different aspects where each aspect is individually presented within its appropriate subsection.

A. Agile vs. Non-Agile

In the remainder of this section, we will use the term "*agile respondents*" to refer to the group of respondents who claim that agile is their current development process. Similarly, we will use the term "*non-agile respondents*" to refer to the group of respondents that claim to use any other development process (e.g., waterfall, adaptive, ad-hoc, etc.).

TABLE III. Survey data for Agile vs. Non-Agile

Interaction with customers/business people should be for capturing the requirements at the beginning of the project and then for acceptance testing at the end of the project								
Current practice	Agile	1	0	4	1	2	1	2
	Non-Agile	1	5	3	10	12	4	1
My preference	Agile	3	2	2	0	1	1	2
	Non-Agile	4	5	4	4	6	11	2
Customers/ business people should be discouraged from changing requirements once they are specified								
Current practice	Agile	4	0	3	2	1	1	1
	Non-Agile	1	4	6	10	10	3	1
My preference	Agile	4	2	3	2	0	1	0
	Non-Agile	1	2	9	5	9	8	2
Once a piece of code starts working, it should rarely be modified								
Current practice	Agile	3	1	3	2	1	0	0
	Non-Agile	2	4	6	13	6	4	1
My preference	Agile	3	2	4	1	0	0	0
	Non-Agile	0	9	8	7	4	6	1
Regular changes to working code should be encouraged if they improve the code in some way (e.g. its design, its structure etc.)								
Current practice	Agile	0	0	0	2	4	3	1
	Non-Agile	1	4	6	13	8	2	0
My preference	Agile	0	0	0	0	2	6	2
	Non-Agile	0	1	5	8	11	7	3
Test cases should be written before writing code								
Current practice	Agile	1	2	4	1	2	0	0
	Non-Agile	1	7	11	8	5	3	0
My preference	Agile	1	1	1	0	3	4	0
	Non-Agile	2	0	5	6	9	12	1

Regular interaction with customers is a central theme in agile development [1] and it can as a benefit provide a continuous assistance in creating and validating acceptance tests. However, looking at the responses from the agile respondents (see TABLE III.), there are (clear) indications that this agile practice is not followed to the extent they would prefer. To some extent, customer interaction is limited to elicitation of requirement and acceptance testing. This is true also for the non-agile respondents, but they are, to a larger extent, happy with the current practice.

The agile respondents are not averse to changes, especially compared to the non-agile respondents, who have a slight tendency to discourage customers from changing requirements. Moreover, in the preferred practice, the agile respondents would like to be even less restrictive to change, whereas the non-agile respondents would like to discourage customers in changing requirements even more.

Changing working software is, as expected, favoured and even more preferred among the agile respondents. Surprisingly, the non-agile respondents do not mind such changes, even though they are less enthusiastic than the agile respondents.

Changes to working software with the specific purpose of improving the structure of the code (i.e., refactoring) is highly preferred among both groups, but the agile respondents appear to be following this practice to a significantly higher extent (albeit not quite at the desired level). It could probably be claimed that the high acceptance of changes to working code is a result of confidence provided by the regression suites created by a test-driven development practice [6]. However, this is not a conclusion we can draw solely based on our data.

Test driven development (TDD) is one of the most prominent practices used in agile development. However, our respondents, especially the ones working in agile processes, are leaning towards disagreement that this practice is in place at their current organisation. Both groups are agreeing their preference is to use this practice to a higher extent.

As a summary, we can observe two facts from this data:

(1) Respondents working in agile processes are not happy with current test first practice (possibly because they are in an early phase of adoption)

(2) Non-agile respondents are unknowingly following agile testing practices.

B. Distributed vs. Non-distributed

By answering the question if all of the team members are collocated in one building, responders are grouped into distributed and non-distributed categories. Questions were asked only to responders who claim to have some testing activities at their work, i.e., the *distributed* and the *non-distributed* respondents make up disjoint subsets of the *testers* category. Collected data is presented in TABLE IV.

TABLE IV. SURVEY DATA FOR DISTRIBUTED DEVELOPMENT

Please indicate how strongly you agree or disagree with the following statements with respect to your testing experience in current organisation								
We never have to wait for source code in order to start the testing process	Distributed	3	2	4	1	3	3	2
	Non distributed	0	3	4	7	2	0	2
The necessary infrastructure for executing test cases is always in place	Distributed	0	2	7	0	6	2	1
	Non distributed	0	5	7	3	1	0	2
There are no changes done on code during integration testing	Distributed	2	5	3	0	8	0	0
	Non distributed	1	3	10	2	0	1	1
During integration testing, I do not mind code to be changed while I am testing it	Distributed	2	1	9	1	5	0	0
	Non distributed	1	0	4	6	6	1	0

When it comes to not having source code available on time to start the testing process, testers working in distributed environments give very diverse answers. In average, they can be considered the same, as most of non-distributed responders, who claimed they neither agree nor disagree this problem exist in their organisation. Since we expected testers working in distributed environments to experience this problem in higher extend, we can only claim that working in distributed development is not a main cause of delays in source code delivery within our respondents.

Not having a proper infrastructure for testing in place could potentially slowdown the testing process. We expected this in particular to be a problem for distributed development, and even though answers were again diverse they were very close to neither agree nor disagree that problem with missing infrastructure exists. However, non-distributed respondents to a slightly higher extent indicated to experience the problem of not having necessary infrastructure in place. Again we could claim within our responders that the problem with not having a proper infrastructure for testing is not directly related to using distributed development process.

Changes done on code could produce further delays and introduce complexity during integration testing. One would naturally expect this to be in much higher degree present in distributed development rather than non-distributed. Our data shows this is not entirely true for our respondents. Both groups tend to disagree there are no changes on code during integration testing with slightly higher level of disagreement within non-distributed respondents. On the other hand, respondents working in distributed development prefer to disagree that they do not mind code to be changed while they are testing it during integration. Non-distributed respondents preferences point out to neither agreement nor disagreement with this statement.

We expected to recognize potential challenges with testing related to distributed development by analysing our data, but as presented in this section, we did not find any such problems among our respondents.

C. Domain

In the *domain* categorization, respondents were categorized according to answers to the following question: *"The software we build in our project is:"*. Possible answers were "*Desktop*", "*Web*" or "*Embedded*". In contrast to the other respondent categorizations in this paper, the domain categories are non-mutually exclusive. Hence, if respondents are working in companies developing products in several domains, they are included in all these domain categories. Survey data for the domain categorization is presented in TABLE V.

TABLE V. SURVEY DATA FOR APPLICATION DOMAIN

Please indicate how strongly you agree or disagree with the following statements with respect to your testing experience in current organization:								
The necessary infrastructure for executing test cases is always in place	Desktop	0	2	4	2	0	2	2
	Web	0	2	4	0	5	2	0
	Embedded	0	4	6	1	2	2	0

I have enough time to test the software before its deployment	Desktop	1	3	0	3	3	2	0	
	Web	0	1	5	1	4	2	0	
	Embedded	0	2	5	3	2	3	0	
In our project we use the following testing types									
Unit testing	Desktop	1	1	0	0	0	2	0	1
	Web	0	1	0	1	0	2	3	4
	Embedded	1	2	1	2	1	3	0	2
Functional black-box testing of the whole system	Desktop	1	0	1	0	2	0	1	1
	Web	2	0	0	1	3	0	4	1
	Embedded	1	0	0	0	3	1	2	5
Performance testing (including load and stress testing)	Desktop	1	0	2	0	0	0	1	2
	Web	2	1	0	0	1	1	2	4
	Embedded	1	0	1	1	2	2	3	2
In my opinion, the ideal level for each of the following testing types in our project should be									
Unit testing	Desktop	0	1	0	0	0	1	0	4
	Web	0	1	0	0	1	1	2	6
	Embedded	1	2	1	1	0	1	3	3
Functional black-box testing of the whole system	Desktop	1	0	0	0	0	2	1	2
	Web	1	0	0	0	0	4	2	4
	Embedded	1	0	0	0	0	2	2	7
Performance testing (including load and stress testing)	Desktop	1	0	1	0	0	2	0	2
	Web	1	0	0	1	1	3	2	3
	Embedded	1	0	0	0	3	2	0	6

In the domain categorization, our expectation was to find that a lack of testing infrastructure would cause problems for testing particularly in the embedded system domain, mainly due to problems of hardware/software co-design and hardware availability in early stages of development. We further believed that this would negatively affect the time available for testing. Moreover, as commonly stated (e.g., in [7]), we hypothesized that development of web and desktop software would, to a larger extent, be influenced by lightweight and agile methods, whereas development of industrial and embedded software would typically follow a more traditional, plan-driven process. There is, however, no support for the latter assumption in the data, as development method seems to be independent of software domain among our respondents.

We do note a more prominent lack of availability of a proper testing infrastructure in the embedded system domain, but not fully to the extent we might have expected. Furthermore, it is worth mentioning that this deficiency does not seem to significantly affect the experienced sufficiency of time available for testing, which is quite equal between the domains.

Regarding the importance of different testing types, in the current practice, web development seems to put a large emphasis on unit testing, whereas embedded system development to a higher degree focuses on functional system testing. In the desktop system development category of respondents, there is no clear indication of a coherent current practice.

Comparing the current practice with what is considered the ideal practice, there are a few noteworthy discrepancies. Generally, among all categories of respondents, there is a preference towards more rigorous testing at all levels, particularly visible in the functional system-level testing. Notable is also the degree in which embedded system developers would like to increase load and stress testing, a practice where they feel that the current level of practice is insufficient.

D. *Safety-criticality*

Prior to the analysis of respondent data, we expected safety-critical respondents to lean towards more traditional types of development and testing in the current practice, but we were curious and more hesitant regarding their preferred practice.

TABLE VI. SURVEY DATA FOR SAFETY-CRITICALITY

Interaction with customers/business people should be for capturing the requirements at the beginning of the project and then for acceptance testing at the end of the project								
Current practice	Safety-critical	0	0	0	2	2	3	1
	Non-safety-critical	2	4	7	8	12	2	2
My preference	Safety-critical	0	1	0	0	0	5	2
	Non-safety-critical	7	5	6	3	7	7	2
Customers/ business people should be discouraged from changing requirements once they are specified								
Current practice	Safety-critical	2	0	2	2	2	0	0
	Non-safety-critical	3	3	7	10	9	4	1
My preference	Safety-critical	3	0	3	0	1	2	0
	Non-safety-critical	3	3	9	7	8	7	1
Test cases should be written before writing code								
Current practice	Safety-critical	1	1	4	1	1	0	0
	Non-safety-critical	1	8	9	8	6	3	0
My preference	Safety-critical	1	0	3	2	1	1	0
	Non-safety-critical	2	1	2	3	11	15	1

When it comes to customer involvement, there is a significant difference between the safety-critical respondents, and the non-safety-critical respondents (see TABLE VI.). The safety-critical respondents to a large extent limit customer interaction to requirements elicitation in the beginning of the project, and acceptance testing at the end of the project. This is something that cannot be seen in the group of non-safety-critical respondents, where the current practice varies. Interestingly, the safety critical respondent preference is to further decrease customer involvement, whereas most non-safety critical respondents would prefer an increase.

Discouraging customers/business people to change requirements is a practice which both groups neither agree nor disagree to currently exists in their organisation. We expected safety-critical environment to be more resistant to change than non-safety-critical respondents, but within our respondents this is not a case. Interestingly, the preference of both groups does not change significantly from the current practice. We could say that changing requirements is not something our respondents would agree easily with, but it is rather something they must accept, regardless of software criticality.

Both the safety-critical and the non-safety-critical categories of respondents state that writing test cases before writing code is mostly not considered as the current practice. However, while the non-safety-critical respondents seem quite willing to change this situation, the safety-critical respondents show no interest as a group in changing towards a more test-driven development. This is noteworthy considering the fact that empirical studies seem to ascribe test-driven developed code a high external code quality [2][3][4]. For fairness sake, it is not trivial to see how such a practice would affect, and be affected by, other specific aspects of safety-critical system development, e.g., fulfilment of safety certification standards.

E. *Testers vs. Non-Testers*

Based on responders' answers to question on how often they perform testing activities, we grouped their response into two categories: Testers and Non-testers. We expected to have insights into differences on how testing related activities are seen from these roles. Data is presented in TABLE VII.

TABLE VII. SURVEY DATA FOR TESTERS VS. NON-TESTERS

Programming should start only after the design is completed								
Current practice	Testers	2	6	10	9	10	1	0
	Non-Testers	1	2	1	2	2	1	0
My preference	Testers	4	5	6	7	5	9	2
	Non-Testers	1	0	1	1	2	4	0
The main focus of the team should be to get the code to work								
Current practice	Testers	0	1	3	6	11	13	4
	Non-Testers	0	2	1	1	3	0	1
My preference	Testers	0	2	6	5	6	13	6
	Non-Testers	0	2	1	2	2	2	0
The main focus of the team should be on the production of all artefacts (e.g. design documents, requirements documents) not just code								
Current practice	Testers	1	3	5	9	16	4	0
	Non-Testers	0	0	3	3	3	0	0
My preference	Testers	1	2	3	7	10	12	3
	Non-Testers	0	0	1	1	1	4	2
Once a piece of code starts working, it should rarely be modified								
Current practice	Testers	5	4	7	13	3	4	0
	Non-Testers	0	1	2	2	4	0	0
My preference	Testers	3	9	11	7	2	3	1
	Non-Testers	0	2	1	1	2	3	0

Testers and non-testers opinions whether programming should start only after the design is completed, do not differ significantly. Both groups are stating this is, to some extent, present in their current practice but importantly to notice, both groups equally point out preference on having this practice in place. Similar results can be seen in question if the main focus of the team should be on the production of all artefacts and not just code. Neither testers nor non-testers agree or disagree this practice exist in their current organisations. However, both group preferences show high level of agreement that team focus should be tailored towards generating all artefacts of software development.

Question if the main focus of the team should be to get the code to work show us some difference in group opinions. Testers think that this philosophy is slightly present in their current organisation, while non-testers tend to disagree. Both groups seem satisfied with their current practice since preference on this idea does not change. Another philosophy was investigated by a question on if once a piece of code starts working, it should rarely be modified. Here, testers show tendency to disagree that this approach exist in their current organisation, while non-testers neither agree nor disagree with it. Interestingly, testers' preferences are to further disagree with this practice, while non-testers show some level of agreement this practice should exist in organisations.

IV. TECHNIQUES AND TOOLS

Respondents, who previously stated to perform testing activities at work, were additionally asked questions regarding tools and techniques currently in use within their organisation. Those questions did not have any predefined answers since we expected the respondents to provide us with inside information about existing testing practices. We expect that an appropriate grouping of information (for e.g.. domain-wise) could be beneficial for new practitioners. TABLE VIII. and TABLE IX. present the respondents' answers followed by our qualitative analysis of the data.

TABLE VIII. RESPONDENT ANSWERS ON TECHNIQUES IN USE

Which testing technique do you use in your organisation? (if you are not sure of the name of the technique, try to explain in short how you perform testing)
Unit testing, integration testing and functional testing.
Low level: Lint, Code coverage, Manual code review. High level: Integration test, Regression test (to verify legacy functionality), Function test (verify new stuff), System test (from an end user perspective)
White box unit testing
unit testing w test coverage strategies (all statements, black-box behaviour) system testing with real hardware done by or together with our customer
No automated testing, only interactive. Though, we have tools which measures coverage as well as performance bottlenecks.
unit testing by developers, per use case manual testing done by testers
No technique as we just do prototypes. We don't test it towards test documents, since just on user tests.

Module test, integration test on dedicated hardware, test on complete machine
1. Manually develop and debug using whatever equipment available using PC/Windows or actual target hardware. 2. Module testing on workstation platform using PC/Windows in a repetitive form. 3. Manual repetitive integration tests on actual target hardware platform. 4. Automated tests on actual hardware and/or system. (not all tests use have automated test cases)
White-box (developers testing their code using debuggers) Black-box (on system level) Unit testing (not so common) Automatic testing (no so common)
Limited regression testing, limited automated user interface testing
It varies from project to project. My current project writes unit tests at the same time as the code, and different people do system testing.
Ad hoc testing, i.e. testing only specific functionality rather than full regression testing for each release.
Testframe
Vast test automation, explorative testing, black-box - white-box, etc, etc...
manual testing
Unit Testing and Integration Testing (White box) by the development team. Black Box (System Testing and Performance Testing) by an independent Test Team
Unit tests, integration & regression tests.
Use case testing State transition testing Classification tree method Boundary value analysis
Ad hoc
Different in different projects.
Acceptance testing against functional and non-functional requirements, creating system test automation in business value order. Unit testing and module testing done by developers.
User-story approach, old-fashioned test-case approach, exploratory testing, test automation, TDD
Component testing done by designers during development. Function testing before delivery to integration branch. Integration testing. System/load testing.

Based on the provided responses (shown in TABLE VIII.) we notice, as expected, that it is very common for organisations to have defined levels of testing. Those levels are usually unit, integration and system level testing. Besides having testers, testing activities are also performed by developers. In most cases, unit level testing using a white box approach is a responsibility of developers, whereas integration and system level testing are done by dedicated testers performed mostly as black box testing. Interestingly, we have not found a big presence of automation effort in testing.

TABLE IX. RESPONDENT ANSWERS ON TOOLS IN USE

Do you use any tools for testing within your organisation? Please provide us with their names:
Not any specific tool.
Expect/TCL, Jcat (java based tool), Perl. Also some proprietary testing platforms based on previously named tools.
NUnit
internally developed tool, simple script on top of a CAN & I/O simulator and same scripts using real CAN & I/O when software is downloaded on target.
Two tools named something with "coverage" and "perform" (cannot remember the company behind)
JUnit, JEmitter
PC-Lint and/or Programming Research QA C for static analysis, MSDevStudio for code coverage analysis of test cases, homebrewed testing harness suited for the current development tools, RTOS supported functions for timing analysis. JTAG/BDM-debuggers for on-target testing. National Instruments LabView using automated test cases derived from requirement tools.
TestComplete (tool for automatic tests)
Yes. Do not know.
Again it varies from project to project. I think my current project uses JUnit, but I have no time to get involved.
Not personally.
code coverage, JUnit(UnitTest)
change control, bug tracking, test case management, etc., etc.
Proprietary in most cases.
Check, valgrind
tcl/expect, Test-RT
Test Director CTE Test execution tools (self-made, using Labview and Perl) NUnit for developer tests
No
Our proprietary test automation system
In-house test automation
TTCN. Load/traffic generators (not sure of name, SipP?).

With respect to tools used as support for testing (shown in TABLE IX.), from respondent answers we can notice that there is utilization of both open source and proprietary tools. However, open source testing tools seem to be mostly used for unit testing whereas for higher level of testing, a proprietary tool is in place. Even a few in-house developed testing tools are stated in the answers.

V. SATISFACTION OF CURRENT PRACTICE

Our third and final analysis in this paper is a quantitative analysis on the satisfaction of current testing practices. If one agrees that knowledge of required process improvements is, to some extent, intrinsic within software development organisations, this information may provide valuable guidelines for future research directions in this area. Below, we will discuss *dissatisfaction* of current practice rather than *satisfaction*. This might seem overly negative, but is rather an effect of measuring the absolute value of the difference between the perceived current and the preferred practices. Hence, a high value indicates a high degree of dissatisfaction. Needless to say, a low degree of dissatisfaction equals a high degree of satisfaction.

Here, individual dissatisfaction of current testing practices is defined as the mean absolute difference between the perceived current practice, and the preferred practice for all testing-related questions for a single respondent. Dissatisfaction among a category of respondents is defined as the mean of all individual dissatisfactions for the respondents within that category. Formally, given the set R

of respondents in a particular category, and the set Q of test-related questions, the dissatisfaction $d_{Q,R}$ is defined by

$$d_{Q,R} = \frac{1}{|Q||R|} \sum_{q \in Q} \sum_{r \in R} |c_{q,r} - p_{q,r}|$$

where, $c_{q,r}$ and $p_{q,r}$ refer to the current and the preferred practice of question q as reported by the respondent r.

A. Satisfaction within Different Categories of Respondents

For the categorization used in Section III, the dissatisfaction results are shown in TABLE X.

TABLE X. DISSATISFACTION WITHIN CATEGORIES OF RESPONDENTS

Respondent category	***Dissatisfaction***
Safety-critical respondents	0.660
Agile respondents	0.728
Desktop respondents	0.821
Embedded respondents	0.875
Distributed respondents	0.880
Web-based respondents	0.910
Testers	0.931
Non-testers	0.933
Non-safety-critical respondents	0.983
Non-agile respondents	0.995
Non-distributed respondents	1.019

As can be seen in the table, the differences between the categories are not remarkably large. The difference between the most dissatisfied category of respondents (non-distributed) and the most satisfied category of respondents (safety-critical) is 0.359, corresponding to little over one third of a grade per question and respondent on a 7-grade scale. Nevertheless, the safety-critical respondents stand out as the most satisfied categories of respondents. This can possibly be attributed to the fact that the training levels in such organisations are very high and focused which make them fully believe in and be confident of the activities and practices. Further investigation will be necessary to confirm that such a conclusion is justifiable from a technological point of view. It is also worth mentioning that the satisfaction of current testing practice among respondents doing distributed development is actually higher than that of respondents doing non-distributed development. The obvious assumption would be that distributed development puts additional strains on the efficiency of testing. A possible explanation of these results might be that a distribution of development enforces the software development organization to be more conscious about its testing strategy.

B. Satisfaction with Particular Testing Practices

In order to analyse the dissatisfaction with regards to a particular practice, denoted by $d_{q,R}$, we make use of a special case of the above equation, where q is the question on the practice of interest and $Q = \{q\}$.

$$d_{q,R} = \frac{1}{|R|} \sum_{r \in R} |c_{q,r} - p_{q,r}|$$

It should also be noted that a high dissatisfaction in a question only tells us that there is a large difference between the current and the preferred practice. It does not explicitly tell us the nature of this dissatisfaction, nor does it mean that respondents agree on whether the practice queried on should be increased or decreased. A prime example of this is the question "*Once a piece of code starts working, it should rarely be modified*", which has a dissatisfaction of 0.935, which is quite high in relation to the total data set. However, taking into account whether the respondents want an increase or a decrease of the practice (mathematically, this is equivalent to disregarding the absolute value operation on the difference between current and preferred practice), we end up with a value of 0.109, indicating that there is almost an equal desire to increase the practice as it is to decrease it, among the set of respondents.

TABLE XI. QUESTIONS WITH THE LOWEST DEGREE OF DISSATISFACTION

Question	***Dissatisfaction***
Changing working code should not be encouraged but cannot be prevented	0.500
Testing should be a defined phase in project development	0.587
Procedures and processes should be allowed to be changed often if the change brings in an improvement	0.652
Project planning should be incremental, one iteration at a time	0.681
How much functionality is in the current working code should be the sole criteria for determining progress of the project	0.696

In TABLE XI. , the five questions with the lowest degree of dissatisfaction are presented. Notably, two out of the three questions with the lowest degree of dissatisfaction concern changes to working code for the purpose of quality improvement or customer satisfaction. Considering the fact that these statements are mostly agreed to by the respondents, it seems to be generally accepted, both in theory and in practice, that changes are inevitable in the software development process, even though they may pose difficulties.

TABLE XII. QUESTIONS WITH THE HIGHEST DEGREE OF DISSATISFACTION

Question	***Dissatisfaction***
Test cases should be written before writing code	1.609
Programming should start only after the design is completed	1.271

Question	***Dissatisfaction***
Comprehensive documentation should be an essential part of software development	1.250
There should be general guidelines and principles for software development but not detailed rules	1.204
Management should encourage regular interaction between developers and customers/business people	1.200

TABLE XII. displays the five statements with the highest degree of dissatisfaction. Out of all the queried practices, test-driven development seems to be the practice where the difference between the preferred practice and the current practice is most significant. Further analysis of the questionnaire data reveals that the cause of this dissatisfaction is that test-driven development is not practically used to the extent that is desired by the respondents. As already shown in Section III, this is true both for agile- and non-agile respondents.

The question *"Programming should start only after the design is completed",* is notable not only because of its high degree of dissatisfaction, but also that because it polarized respondents to such an extent. Seven respondents recognized this as the current practice, and would like a decrease. On the other hand, twenty respondents felt that the programming starts too soon and should be postponed until the design is completed.

For the remaining three questions in the table, the dissatisfaction in the practices is due to the fact that the practices are preferred, but not followed.

VI. CONCLUSION

In this paper we present our analysis results from an industrial questionnaire survey on the current software development practices and preferences, specifically in relation to testing. The survey has unique features such as strategic embedding of multi-purpose questions and categorisation of respondents on contemporary aspects which enable us to gain qualitative insights. The survey in addition to confirming some popular beliefs also lists several noteworthy findings from the perspectives of respondent categories such as safety-criticality, agility, distribution of development, and application domain. These findings clearly depict negative discrepancies between the current practices and the perceptions of the respondents, thus meeting our research objective presented in the introduction.

One of the noteworthy testing research directions from an industrial perspective seems to be test driven development as indicated by the results of the survey. Our ongoing research work attempts to refine these findings through directed interviews as well as through further investigations in a wider context. We are also working on the definition of a methodology for dynamically incorporating such findings in the management decisions on strategic challenges such as introduction of new technologies, processes, and effort allocations in relation to testing.

ACKNOWLEDGMENT

This work was supported by the VINNOVA through the ITEA2 project FLEXI. The authors want to express their thanks to all the questionnaire respondents and the people involved in earlier phases of this research.

REFERENCES

[1] G. K. Hanssen and T. E. Faegri, "Agile customer engagement: a longitudinal qualitative case study " in Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, 2006, pp. 164-173.

[2] L. Williams, E. M. Maximilien, and M. Vouk, "Test-driven development as a defect-reduction practice," in *Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on*, 2003, pp. 34-45.

[3] A. Marchenko, P. Abrahamsson, and T. Ihme, "Long-Term Effects of Test-Driven Development A Case Study," in *Agile Processes in Software Engineering and Extreme Programming*, 2009, pp. 13-22.

[4] B. George and L. Williams, "A structured experiment of test-driven development," Information and Software Technology, vol. 46, pp. 337-342, 2004.

[5] A. Causevic, I. Krasteva, R. Land, A. S. M. Sajeev, and D. Sundmark An Industrial Survey on Software Process Practices, Preferences and Methods, MRTC report ISSN 1404-3041 ISRN MDH-MRTC-233/2009-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, March, 2009.

[6] K. Beck, Test Driven Development: By Example: Addison-Wesley Longman Publishing Co., Inc., 2002.

[7] J. Ronkainen and P. Abrahamsson, Software Development under Stringent Hardware Constraints: Do Agile Methods Have a Chance? In *Extreme Programming and Agile Processes in Software Engineering*, 1012, Springer, 2003.