Knowledge and mindset in software development – how developers, testers, technical writers and managers differ – a survey

Attila Kovács
attila.kovacs@inf.elte.hu

Kristóf Szabados*
kristof.szabados@ericsson.com

ELTE Eötvös Loránd University
Faculty of Informatics
Budapest, Hungary

Abstract

Creating software products is a complex endeavor requiring the cooperation of people with different skills/knowledge/thinking. Managers, developers, testers and technical writers have to work together to create and deliver software products. These roles might have different views, perceptions, knowledge even in the same project.

In order to understand their commonalities, differences and the evolution of their skills we run a survey that was filled in by 456 professionals working in software development projects.

We have found among others that (1) Internet is one of the most useful source of information for professionals; (2) trial and error is perceived to be more efficient then formal training; (3) testing skills are among the most important ones; (4) there are little differences in people's way of thinking compared by roles, by size of employing company or by experience levels; (5) at the same time, larger companies seem to be more efficient and their experts are better at monitoring and testing new technologies/methods; (6) interestingly, although most companies support the improvement of internal quality, most respondents have very limited knowledge or are not concerned of anti-patterns.

1 Introduction

Meet Joe, a software developer working in a team of 4-7 members. His friends, Jennifer the manager and Ben the tester, both collaborate with 30+ people. Finally Alice, a technical writer, is again part of a team of 4-7 members.

All of them work at companies employing 1000+ people and have less than 2 years of experience. In their eyes formal training is a waste of time, they learn from the Internet and on-the-job. They believe that developer and tester mindsets are both important. Although their companies support them in creating quality products, they mostly don't know or don't care about internal quality. According to our survey they represent the largest groups of people in their field, working in software development projects at companies present in Hungary in the year 2015.

In: E. Vatai (ed.): Proceedings of the 11th Joint Conference on Mathematics and Computer Science, Eger, Hungary, 20th – 22nd of May, 2016, published at http://ceur-ws.org

We surveyed individuals working in software development projects. We wished to understand how the knowledge of IT employees differs having various roles (manager, developer, tester, technical writer), how they gain new knowledge, how they vary in thinking about their processes and anti-patterns in software development. This paper presents the results of the survey focusing on roles, experience levels and the size of the companies respondents were working for.

Our research questions are:

- *RQ1* How well known are the techniques of different fields?
- *RQ2* How important are the different mindsets?
- *RQ3* What are the main sources of new knowledge?
- *RQ4* How useful are the several knowledge gaining methods in daily work?
- *RQ5* How different is the way of thinking in the various roles?
- *RQ6* How are anti-patterns perceived?
- *RQ7* Are people motivated and supported to resolve anti-patterns?
- *RQ8* How does the size of the company or team organization impact people's knowledge, thinking and perception of anti-patterns?
- *RQ9* How does experience level impact people's knowledge, thinking and perception of anti-patterns?

This paper is organized as follows. In Section 2 we present earlier works related to anti-patterns in software development. Section 3 details the methodology used to create and run the survey. Section 4 shows the generic answers and details on each role group. Later we address how the same questions are related to the size of company in section 5 and experience level of respondents in section 6. Section 7 summarizes our results and observations. Finally, section 8 deals with the validity of our results.

2 Previous works

2.1 Anti-patterns

Research into anti-patterns started with Fowler's work [1] in software development. Fowler used the term *code smells* to describe issues in the source code that might indicate architectural problems or misunderstandings, issues which are very hard to detect. In the following sections we use the term *anti-pattern* as a more general form of talking about *code smells*. This is necessary to describe the similar phenomena present in the fields of management and technical writing, where source code of any kind might not be directly involved.

Since then the initial list of 22 code smells has been extensively extended (see e.g. [2], [3] and [4]).

Knowledge on anti-patterns in testing is found scattered on the Internet. In their blog posts Carr [5] collected 23 anti-patterns of Test Driven Development and Scott [6] published his observations and ideas regarding anti-patterns. Juristo et al. [7] found that more than half of the existing testing knowledge in 2002 was lacking of any formal foundation. Their major conclusion was that the knowledge of testing techniques was very limited at that time. Even the reference book by the International Software Testing Qualifications Board [8] mentions patterns mostly in contexts of testing for a given data pattern, the recognition and handling of anti-patterns is not much covered.

Stamelos et al. [9] observed that management anti-patterns are likely to appear in student's projects and may cause trouble, affecting the final product. Their introduction to anti-patterns shows why these practices are hard to observe: "In Software Project Management, commonly occurring, repeated bad practices are stated as anti-patterns. These practices are used frequently by software companies because they disguise themselves as an effective and efficient way to resolve common problematic situations, hence rendering it difficult for their negative consequences to be identified. As a result, many project failures in software industry can be attributed to the appearance of anti-patterns..." (They cited [10] and [11].)

In the field of technical writing most books teach techniques using structures and patterns (e.g. [12]). Breaking the pattern of alphabetical ordering, sentence structure or using jargon might be recognized as an anti-pattern. Otherwise, we have not found any article or study aimed at discovering anti-patterns in technical writing.

2.2 Previous similar surveys

In 2013 Yamashita et al. [16] conducted a survey finding that 32% of the respondents did not know about code smells, nor did they care. Those who were at least somewhat concerned about code smells indicated difficulties with obtaining organizational support and tooling.

The "State of Testing 2015" survey [17] showed that the demand for testers, who can do more than "just testing", is increasing. 81.5% of the testers reported to learn their mastery mostly while doing their work, while only 17% on formal trainings.

The "Worldwide Software Testing Practices Report 2015-2016" [19] survey found that organizations use on the job trainings (72.9%), certifications (51.9%) and formal training (46%) to improve the competency of their employees. This survey also found that Agile management techniques (Scrum, Extreme programming, Kanban) are being adopted more often (69.6%) in software development projects.

3 Methodology

In our survey we investigated the knowledge and concerns of people working in software development projects.

3.1 Objectives for information collection

Our main goal was to explore the thinking of software professionals working in different roles, to gain knowledge on how they align with industry-standard processes. The secondary goal was to explore what they know, how they learn, and how they are committed to internal quality.

3.2 Preparation of the survey

To get comparable information from different fields involved in software development we used two control groups. We asked the first control group – at least one person from each target group – to evaluate our questions. They were given the survey with the explicitly stated aim of validating the expressions/sentences. Once the reported issues were corrected we created a second control group. This time participants were asked to fill in the survey on the web form it will appear later in. This was done in order to validate the corrections of earlier mentioned issues and to discover potential problems/technical difficulties with the layout of the survey. The results of the control groups were not included in the final survey.

To reach as many people as possible we created our anonymous survey with Google Forms using a minimum number of open ended questions. To track the knowledge of respondents we used questions with several predefined answers. To track the opinion of respondents we offered scales with five options. At some questions we asked for percentages of time spent with some activity.

3.3 The structure of the survey

We grouped the 47 survey questions (found in the appendix) into 6 sections:

1. "Generic information" established information, regarding the respondent's main role, task and size of their organization.

2. "Familiarity with different techniques" contained specific questions related to the four main targeted role groups to understand the actual knowledge of participants.

3. "Gaining new knowledge" collected information on how and from where participants gather new knowledge to improve their existing skills.

4. "Process and methodology related questions" assessed how many participants follow the industry-standard methods in their work.

5. "Anti-patterns" contained questions on how the participants are committed on the internal quality of their work.

6. "Static analysis and traceability" contained questions on static analysis tools, reviews, traceability issues.

3.4 Spreading the Survey

Exploiting our social networks we contacted IT people from several companies (performing software development) with offices mainly in Hungary and asked them to fill in the survey (for example Ericsson, LogMeIn, NSN, SAP, NNG, Prezi, GE). We have also contacted several meetup[1] groups to let us advertise our survey on their site: Test & Tea, Hungarian C++ Community, Budapest DevOps Meetup, Freelancers in Budapest. The survey was posted to the Hungarian IT professionals group at `www.linkedin.com`. From the public forums we used `www.hup.hu`[2] and `www.prog.hu`[3].

We have also contacted the `Technical Writers` group of facebook.

4 Results regarding the roles

In total we received 456 responses from several professionals: 39 architects, 8 business operation supporters, 171 developers, 2 executive managers, 10 line managers, 3 manager of managers, 20 project managers, 2 self-employed, 28 team leaders, 28 technical writers and 145 testers.

To make the information processing easy we grouped the roles into four distinct groups: developers (210), testers (145), managers (71) and technical writers (28). At the end we decided to exclude responses from the self-employed respondents. Their answers could not be merged into the other groups as they might do in their daily work all of the tasks of each role group. At the same time we could not analyze their answers separately as that could have compromised their anonymity.

In order to be able to calculate statistics we mapped the "Not required – Required", "Never – Always", "Not concerned – Concerned" terms in the answers to the scale from one to five points.

4.1 Generic information

86% of the respondents work for multi-national companies (85% of developers, 89% of testers, 81% of managers, 96% of technical writers). All but one technical writers responded to work for a multi-national company.

63% of the respondents work for companies having 1000+ employees. The ratio of testers is the highest in 501-1000 employee companies (52%), while the ratio of developers is the highest (70%) in companies employing 10 or fewer people (Fig. 1(a)).

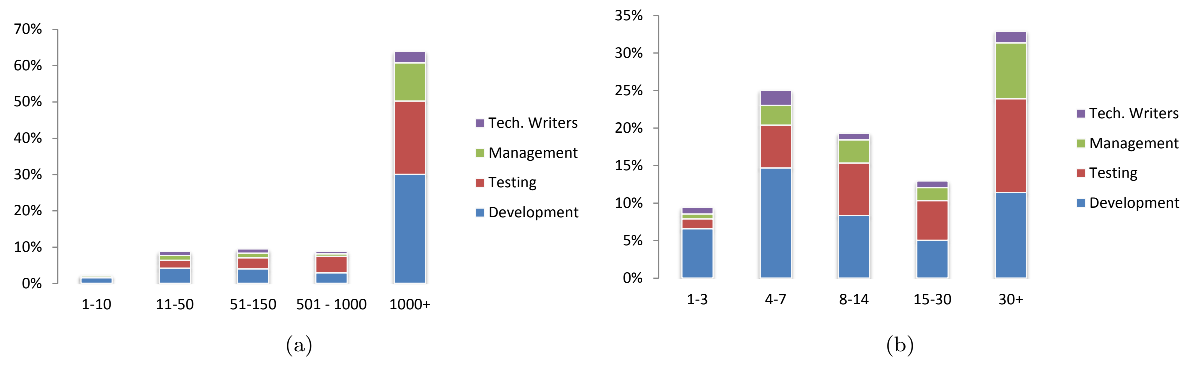


Figure 1: 1(a) Company size distribution the employees are working for; 1(b) Group sizes the employees belong to in their main projects

32% of the respondents work together with more than 30 people in their main project (Fig. 1(b)). The second largest team size is 4-7. Most of the managers (47%) and testers (39%) work together with more than 30 people. Most of the developers (31%) work in projects of team size 4-7, just like technical writers.

51% of the respondents have less than 2 years of experience (29% have 3-5 years, 11% have 6-10 years, 7% have over 10 years of experience). We observed approximately the same ratio in all four groups (except that no technical writers reported to have 6-10 years of experience).

Figure 2 shows the tasks of people reflected to their role in 2014. Developers were developing systems (44% of all respondents), editing code (22%) and doing maintenance work (9%). Testers were testing (79%). Managers

[1]communities organized on www.meetup.com
[2]Hungarian Unix Portal
[3]a web portal claiming to be the largest developer and programmer community in Hungary

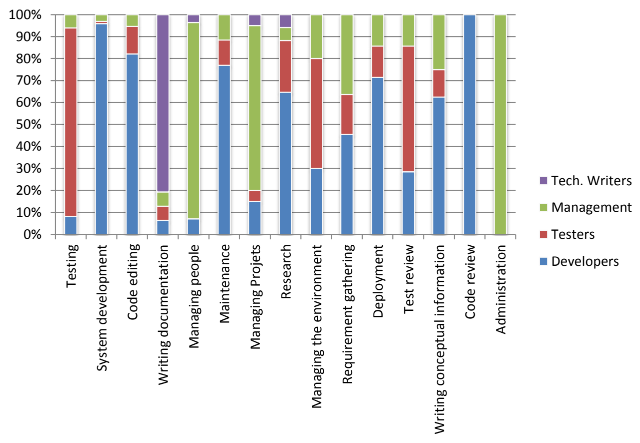


Figure 2: Tasks of the respondents.

managed people (35%) and projects (20%). Technical writers wrote documentation (89%). Only developers did code reviews (1%) as main task, the environment was mostly managed by testers (3%).

As most common additional responsibilities we recorded writing documentation (48%), testing (47%) and code review (43%). Test review and code editing took 4th and 5th place (37%) overall.

The most common secondary tasks were: for developers code review (67%) and testing (30%), for testers test review (67%) and writing documentation (53%), for managers managing people (42%) and administration (38%), for technical writers administration (39%) and product "research" (35%).

4.2 Familiarity with different patterns

Both developer and tester mindsets are very important in software development projects. While testing techniques are well known, development techniques rank as the least known.

The top three known design patterns are (Fig. 3(a)): singleton (55%), iterator (52%) and factory (49%).

The top three known testing patterns are (Fig. 3(b)): function testing (89%), use-case testing (69%) and review (64%).

The top three management methodologies are (Fig. 3(c)): scrum (88%), agile (87%) and waterfall (82%).

The top three technical writer patterns are (Fig. 3(d)): user documentation (64%), system documentation (59%) and review (38%).

In each experience level the ratio of people knowing any given design pattern is similar (Fig. 21(a)).

Developers, testers and managers know approximately the same ratio of testing techniques, management techniques and technical writing techniques.

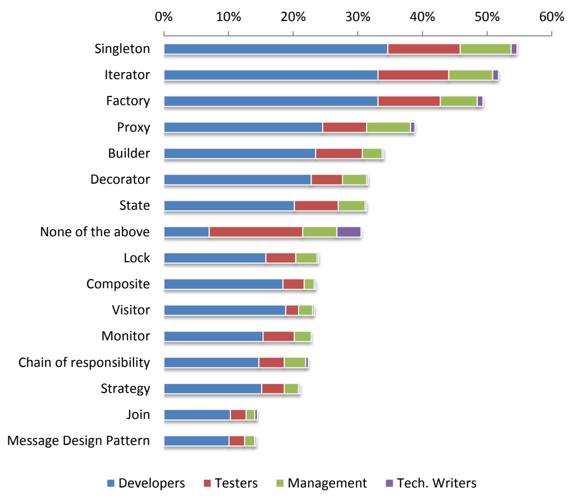
Technical writers know review, walk-through, inspection the most from testing techniques and scrum, agile and waterfall from management techniques. Technical writers have a balanced knowledge, more emphasis on analysis of audience, precise expressions proof-reading and less emphasis on user and system documentation.

Managers concentrate more on focus groups, documentation life-cycle management, and less on user testing and review.

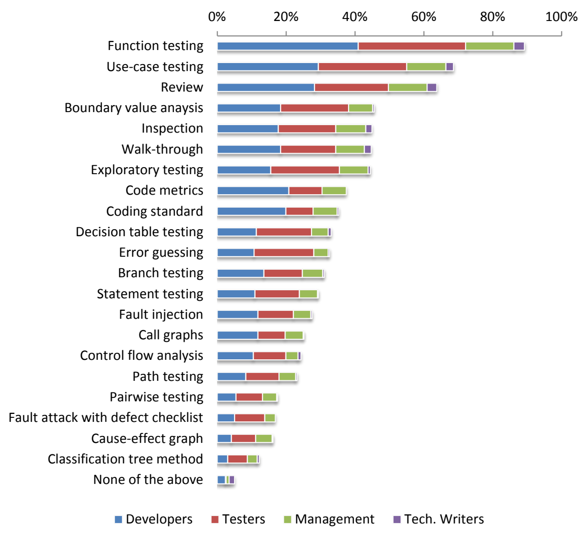
Comparing all patterns we can see that the most known techniques are: Function testing (89%), Scrum (88%), User documentation (64%) and Singleton (55%).

Developer mindset was selected to be important (4-5 points) by all groups (93% developers, 61% testers, 65% managers and 46% technical writers). Testing mindset was selected to be important as well (4-5 points) by all groups (76% developers, 97% testers, 69% managers and 50% technical writers). Technical writer's mindset was selected to be important (4-5 points) mostly for technical writers (13% developers, 36% testers, 24% managers and 96% technical writers). Management mindset was selected to be important (4-5 points) mostly for managers (15% developers, 41% testers, 93% managers and 57% technical writers).

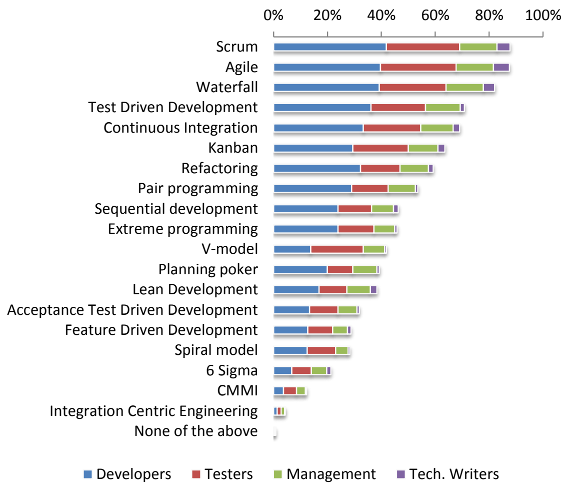
Altogether, the developer and tester mindsets were selected to be the most important in the software projects (Fig. 4(a)). This points to an interesting observation: testing mindset is reported to be important and testing techniques are well known, however, development techniques are the least known, but the mindset was still considered to be one of the most important. Management mindset is considered to be only the 3rd on the



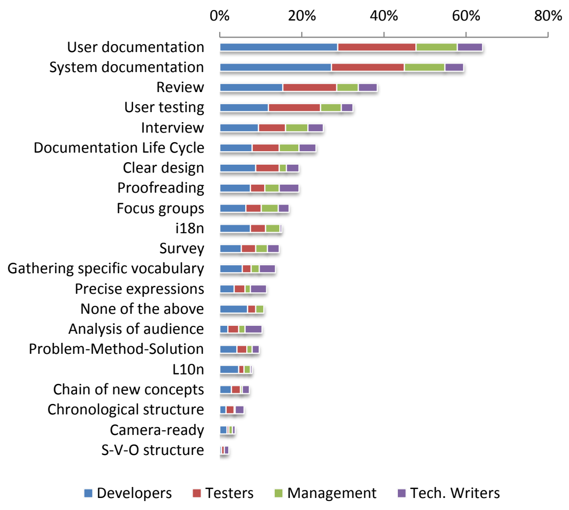
(a) design patterns



(b) testing patterns



(c) management methodologies/patterns



(d) technical writing patterns

Figure 3: Knowledge in the different groups.

importance level, still, some techniques are known by 30% more respondents than the most known development technique.

4.3 Gaining new knowledge

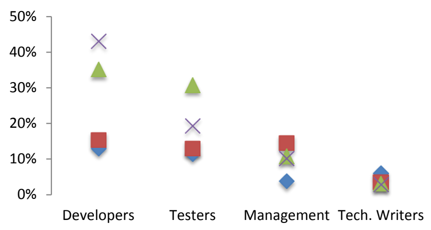
The top three sources of new learning (Fig. 4(b)) were: Internet forums and blogs (82%), colleagues (79%) and books (65%). All 4 groups investigated show similar preferences. These results were repeated in the answer on what resources they have used in the previous year.

Some additional sources for gaining new knowledge (that were not included but remarked in the questionnaire): meetups, online courses, self study.

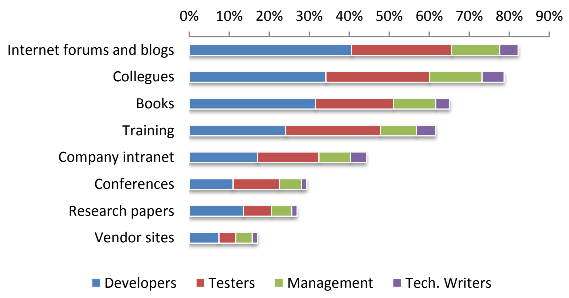
We found (Fig. 5) that all role groups came by approximately the same ratio of knowledge through formal training (24%). However, the maximum ratio were very different: some developers could gain 100% of their knowledge in this way, while for technical writers the maximum was only 50%.

On-the-job training was most useful for managers (41% on average) and least useful for developers (30% on average). In this case the maximum ratio reported was 90% for technical writers, and 100% for all others.

Self study is the main source of knowledge for developers (44% on average), while technical writers use it the least (31% on average).

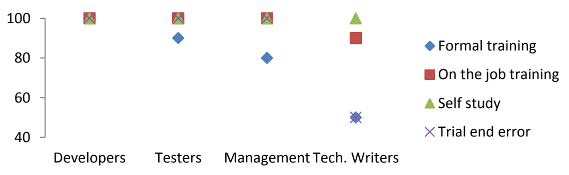


(a) design patterns

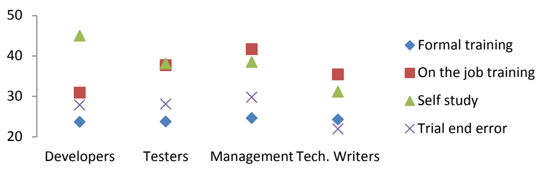


(b) information sources

Figure 4: 4(a) The importance of different mindsets, 4(b) The main sources obtaining new knowledge.



(a) max



(b) avg

Figure 5: The maximum and average values of how knowledge is gained (in percentage).

Trial and error is the source of 27-29% of knowledge for developers, testers and managers (on average). Some of them reported to gain 100% of their knowledge this way. Technical writers gain only 21% of their knowledge in this way (on average), and none of them reported to have gained more than 50%.

Technical writers can rely the least on both formal trainings and learning by trial and error. They might get the same amount of knowledge from these methods, but they can get at most half of their experience in these ways (on average).

Formal trainings are less useful than trial and error based learning, and less people could claim to have learned everything on these ways.

4.4 Process and methodology related questions

To be able to compare the different groups of people participating in software development projects we decided to check how strong their process and methodology orientation is versus an ad-hoc and intuitive practice in their daily work (we call this as "scientific" thinking). We asked whether (1) people are monitoring the world for new ideas and evaluate them critically before inserting into daily practices, (2) they are establishing hypotheses about the target before performing any change when the current situation is assessed, (3) people are able to detect if there is any flaw in the process planning, in the execution of the process or in the results, (4) the flaw is analyzed rigorously.

Results show (scores between 3 and 4) that at most companies it is somewhat important to work in a manner fulfilling strict processes and methodologies in order to see from where and to where tasks and people are heading, to understand where and in what position the work is standing.

When we compared the main roles of the respondents based on their scientific thinking/methods we observed that respondents in development, testing and technical writing show similar values, while managers provided distributed answers (Fig. 6 and Fig. 7). The average standard deviations for the process and methodology related questions (Fig. 7) in descending order: Q28 $(1,14)$, Q23 $(1,13)$, Q24 $(1,12)$, Q25 $(1,11)$, Q30 $(1,1)$, Q35 (1), Q34 (1), Q26 (1), Q32 (1), Q22 (1), Q33 $(0,99)$, Q31 $(0,99)$.

Checking the correlation coefficients between the answers given by people having different roles revealed the following:

- The highest correlation could be found between developers and testers: 0.93 on average.
- Developers and architects way of thinking had the second largest correlation coefficient: 0.90.

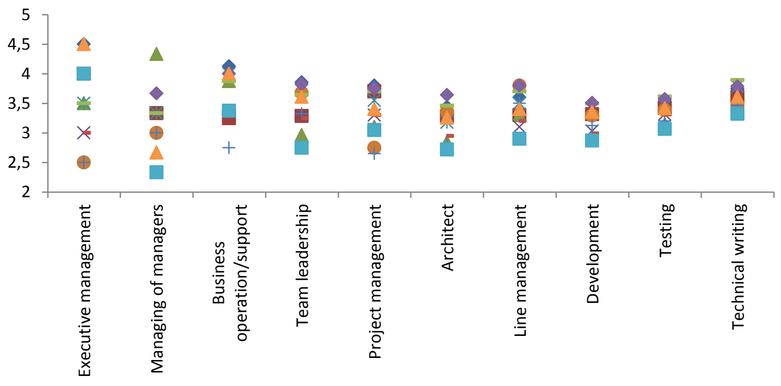


Figure 6: Process and methodology importance depending on different roles (1: least, 5: most, on average)

- The team leadership way of thinking is correlated with (1) development: 0.89, (2) testing: 0.88, (3) line management: 0.85, (4) architect: 0.84.
- The architect way of thinking is correlated with: (1) development: 0.90, (2) testing: 0.89, (3) team leadership: 0.85, (4) line management: 0.82.
- We also observed a correlation of 0.80 between technical writing and testing mindsets.

All other correlation coefficients were below 0.80. The process and methodology orientation of the management (including executive management, managing of managers, business operation/support, project and line management) has little in common with each other and with other roles. In the technical writer's thinking they are the closest to testing (0.80) and development (0.78).

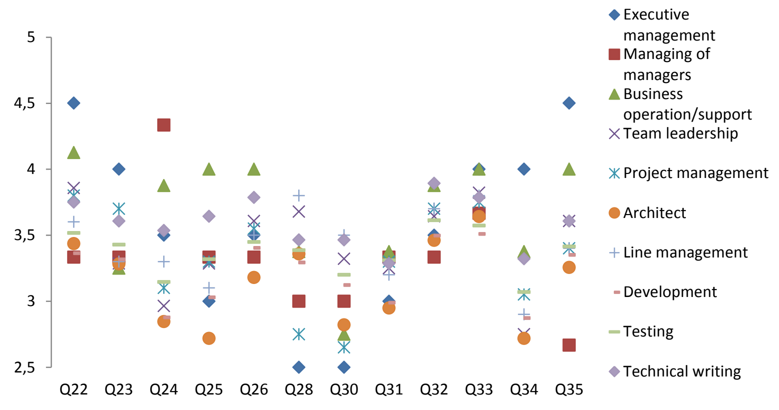
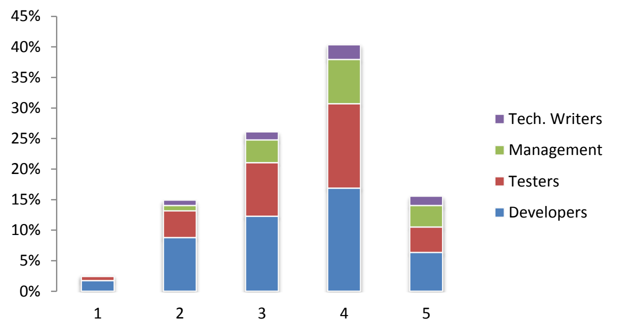


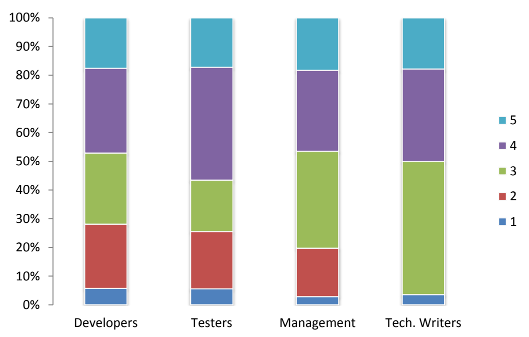
Figure 7: Answers for process and methodology related questions (1: least, 5: most, on average). Q22: technology awareness, Q23: technology introduction, Q24: technology/methodology pilot, Q25: process improvement, Q26, Q28, Q30: process modification, Q31: process introduction, Q32, Q33: process definition, Q34: process performance, Q35: process redesign. (The full questions can be found in the Appendix)

Respondents reported (Fig. 8(a)) that in their company the newest technologies/methodologies are frequently monitored and evaluated (from never to always: 2%, 15%, 26%, 40%, 16%). Mostly managers and technical writers responded with confirmative values (∼70%), but even most of the developers and testers perceived that their organizations perform these tasks (∼50% confirmative answers).

We had similar distribution of answers for the question of how extensively new technologies are tested before introducing them into the organization's life (from never to always: 5%, 20%, 26%, 33%, 19%). We found that developers, managers and technical writers gave 4-5 marks in ∼50%, while testers in ∼60% of the cases (Fig. 8(b)). Technical writers found their tools best tested before introduction.



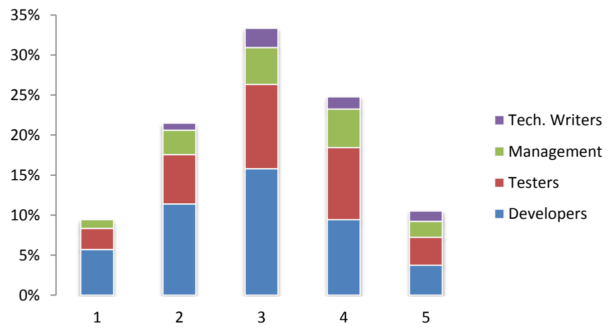
(a) Monitoring and evaluating the newest technologies/methodologies in the company (number of respondent, 1: never, 5: always).



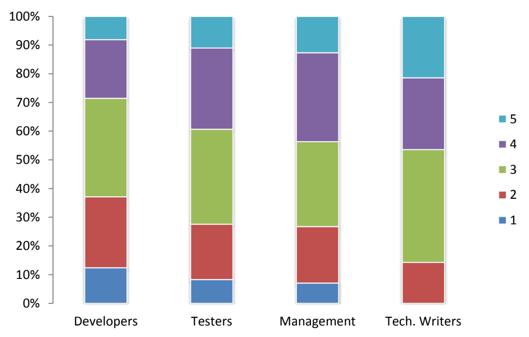
(b) When a new piece of technology/methodology is available, extensive testing is performed before introducing it into the organization's life (1: never, 5: always).

Figure 8: Monitoring and testing new technology before introduction.

The answers for the question "how often testable hypotheses are established before work starts?" were in the middle of the frequency range (Fig. 9). In this case the different groups had very different perceptions. Developers gave the fewest high values and technical writers the most.



(a) by selected value



(b) by role

Figure 9: When a new activity/artifact is defined then sets of hypotheses are established that can be tested before work starts (1: never, 5: always).

When an activity is not done as specified respondents mostly follow a defined process to improve it (Fig. 10). Developers rate their improvement processes the weakest, while technical writers the best.

When the outcome is defective despite all activities done as specified respondents mostly modify their processes. Again, developers gave the lowest values for the frequency of their process modification procedures, while technical writers gave the highest values.

Approximately half of the respondents in all groups reported the ability to detect when someone is idle for long and then follow a defined process to modify or reassign activities. Respondents reported to have been idle 9-15% of their time in 2014 independently from their roles. The maximum idle ratio was almost twice longer for developers and testers, and almost one and half times longer for managers than for technical writers.

Approximately 40% of the developers, testers and managers reported high confirmative values (4-5 points) for being able to detect if someone is overloaded and then being able to follow a defined process in order to modify or reassign activities. The average ratio of being overloaded in 2014 was 24% for developers, 28% for testers and 31% for managers and technical writers. The maximum reported ratio of being overloaded was very high in all groups.

Only 30% of developers are able to redesign their processes if they find that a non-specific activity is needed compared to the ~45% of testers and managers.

In all groups, the respondents were able to detect easily what the next activity is in their processes being actually performed. High values (4-5 points) were given by ~55% of the developers, ~60% of the testers, ~64% of the managers and ~68% of the technical writers. In all groups only ~15% of the respondents gave scores

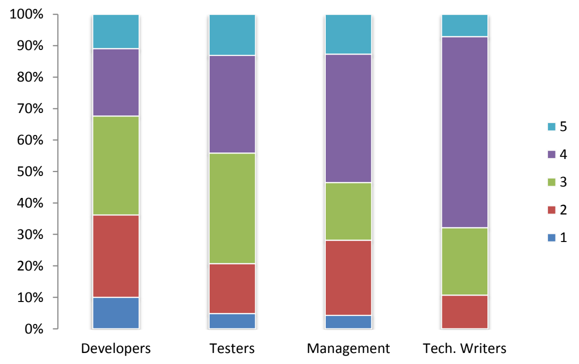


Figure 10: When an activity is not done as specified a defined process is followed in order to improve it (1: never, 5: always).

below 3.

We observed the same ratio for determining who has to perform the next activity in the process.

Only ∼30% of the developers, testers and managers check the current state of affairs rigorously before making a change, compared to 45% of technical writers. Only 5% of the respondents reported that they always asses the current state of affairs before making a change.

When the results are not the ones expected ∼50% of the developers, testers and technical writers check how the change was done and what effects it might had, compared to the 60% of managers.

4.4.1 Greatest deviation from the mean by role

When we look at how people in the different roles rate their processes and methodologies (Fig. 7) we get some interesting insights into how much and where their thinking (perception of their processes) differs. Based on the average values of each role for each question (that fall outside the 3.2 - 4 range):

- Executive Managers believe they are monitoring new technologies (4.5) and carefully testing them before integration (4). The current state is assessed before making a change (4) and if a change has a different effect than expected the reason is checked (4.5). At the same time they believe they are the least likely to identify if someone is idle (2.5) or overloaded (2.5); to find the reason for non-specific activities (3) or improving in case of wrong execution (3).

- Managers of managers believe they set up hypotheses before work starts (4.3), but are least likely to check the reason if the result of a change is different from the expected (2.6).

- Business operation/support believe they asses the current state rigorously (4), know clearly what the next activity in the process is (3.8) and who has to carry out the next activity (4). They try to improve after a bad outcome (4) and modify processes (4). At the same time they are bad at telling who is overloaded (2.75) and testing new technology before introducing it to the processes (3.3).

- Team leaders believe they are bad at finding out who has to carry out the next activity (2.75) and establishing a testable hypotheses before work starts (3).

- Project Managers find it hard to identify idle (2.75) and overloaded persons (2.65). They also don't believe they create testable hypotheses before starting the work (3.1), or assessing current state of affair with rigor (3).

- Architects generally give scores between 2.9 and 3.5. They don't believe to have an improvement process to follow when something goes wrong (2.7), or to assess the current state before making changes (2.7). They also don't believe they create good hypotheses before starting the work (2.8), or find out why a non-specific activity is needed (2.9), or telling if someone is overloaded (2.8).

- Line managers believe they have good processes for telling who is idle (3.8), what the next activity is (3.7) and who has to carry it out (3.78). They don't believe they are assessing the current state before a change (2.9) or follow a process to improve (3.1).

- Developers generally give scores between 3.1 and 3.4. They don't believe they assess the current state (2.87) and establish a hypotheses (2.87) before starting the work. They also don't believe that, when something is not done as specified or some extra activity is need, they follow a process to improve (3) and redesign their processes (3).

- Testers generally give scores between 3.2 and 3.5. They don't believe they assess the current state (3) and establish a hypotheses (3.1) before starting the work. They also don't believe that their team is able to identify overloaded people (3.2).

- Technical writers generally give scores between 3.5 and 4. They believe it is it clear what the next activity is (3.9), who has to carry it out (3.78) and when they defective out in spite of doing everything right they modify their processes (3.78). They least believe they can find out why some non-specific activity is need (3.28) or assess the current state of affair with rigor (3.32).

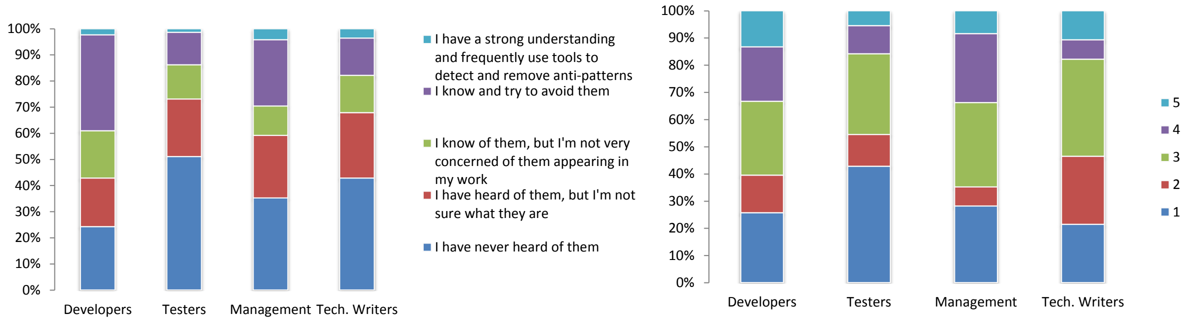
4.5 Anti-patterns

Although most companies support the improvement of internal quality, most respondents have never heard of or are not concerned about anti-patterns.

We have described anti-patterns as "an anti-pattern is a common response to a recurring problem that is usually ineffective and risks being highly counterproductive" in the survey.

35% of the respondents answered to have never heard of them, and 20% to have heard of anti-patterns but not sure what they are. 15% know them, but are not concerned. Only 25% reported trying to avoid them, and 2% reported a strong understanding. Anti-patterns are most understood by developers, and least by testers (Fig. 11).

When checked the question in more detail, we got that 51% of the architects tries to avoid them, 87% of business operations/support have never heard of them or are not sure what they are. 26% of the developers have never heard of them, 19% are not sure what they are, 19% are not concerned, 33% try to avoid them, but only 2% have strong knowledge and use tools to detect and remove them. Line, executive and manager's managers have balanced knowledge (half of them are familiar with anti patterns on some level, half of them not). 75% of project managers have never heard of them, are not sure what they are, or are not concerned. Only 12% of the testers know and try to avoid them and only 1% uses tools for detection and removal.



(a) Familiarity with design anti-patterns.

(b) The severity of taking anti-patterns into consideration (1: least, 5: most).

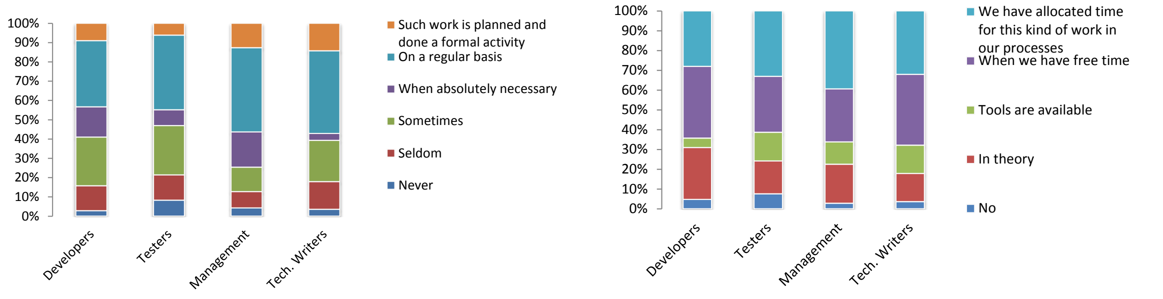
Figure 11: Anti-patterns.

When asked how concerned respondents are about anti-patterns in their product, 31% of them reported to be not concerned and 30% to be mildly concerned. In all role groups at least 20% of the respondents were not concerned at all and only 5-15% were concerned (Fig. 11(b)). Developers (13%) and technical writers (10%) being the most concerned.

The result means that:

- at least 60% of the respondents in all groups are supported by his organization to improve the internal quality of their products (Fig. 12). The ratio is the best (65%) for technical writers.

- at least 40% of the respondents either have pre-planned sessions and work lists for internal quality improvements or correct such issues immediately when they notice them.



(a) The abundance of working on existing products in order to improve their internal quality.

(b) The necessity of working on existing products to improve their internal quality supported by your organization.

Figure 12: The organizations support for internal quality improvement.

- less than 6% have reported to have no organizational support for internal quality improvements.
- less than 7% have reported to have no process for internal quality improvements.

In 2014 most respondents produced low quality results in order to satisfy short term needs 1-5 times (35% 1-2 times, 29% 3-5 times). There were 68 respondents who did not need to give up on quality (Fig. 13), while 11% produced low quality 10+ times. The ratio of no compromises was best among technical writers (21%), followed by developers (18%), testers (13%) and managers (7%).

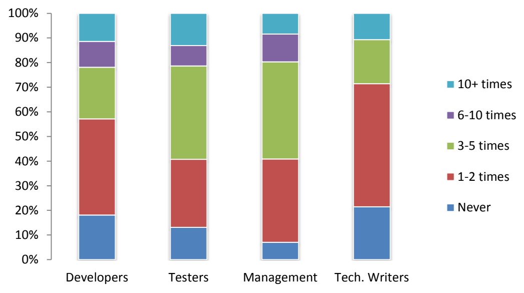


Figure 13: The frequency of producing low quality solutions in order to satisfy short term needs in 2014.

4.6 Static analysis and traceability

Our further analysis shows that most of the found issues are traced back to the early stages of processes and are controlled by static tools, manual code reviews and direct contact to customers.

According to respondents most issues can be traced back to code writing, concept/system design and requirement collection (Fig. 14). Regarding the role groups we had similar rates except that technical writers found task management as a source of problems principally and placed less emphasis on code writing.

Both technical writers and testers placed the most emphasis on the available documentation as the source of problem solving. Most organizations apply tools to statically check adherence to coding standards and to measure metrics (Fig. 15). At this point we observed a clear difference in the roles: developers, testers and managers take static analysis tool supports by approximately the same percentage in their work, but technical writers reported to be less supported in checking coding standards and measuring metrics. They also reported the highest ratio of not being supported by static analysis tools.

We asked furthermore whether manual code reviews are used in internally developed products: 73% answered yes (81% of developers, 67% of testers, 74% of managers and only 39% of technical writers, Fig. 16).

The average time of manual code reviews took 51 minutes for testers and technical writers, 56 minutes for managers and 58 minutes for developers. The maximum time spent with manual reviews was 8 hours for managers and technical writers, 16 hours for developers, and testers could spend up to 24 hours.

By our measurements 40% of the respondents selected to have no direct contact with their users (Fig. 17). After direct email (51%) this option received the second most votes.

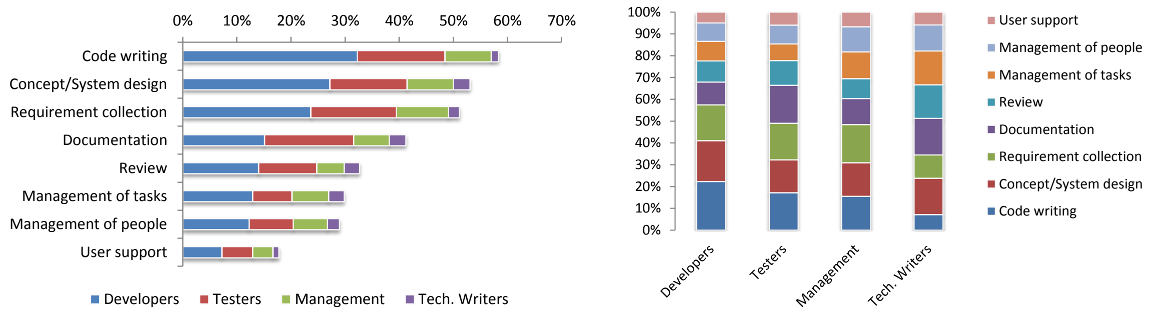


Figure 14: Processes to which the found issues were traced back in 2014.

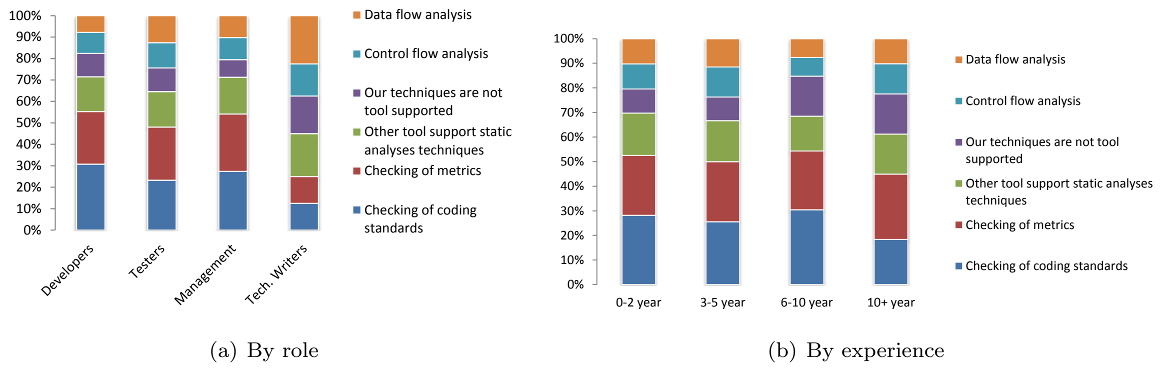


Figure 15: The application of tool supported static analysis techniques (above roles:, below: experiences).

Some respondents mentioned that customer support and issue tracking tools are "the" direct contact to users.

The question "How do you judge if a specification is out-of-date?" was offered to the respondents in order to describe the situation with their own words. 30% of the respondents gave any answer. 4% categorized as "do not care", 2% answered to check the version number of the appropriate document, 1.9% decided to verify the date of the last modification, 2.5% would ask for help to decide. 1% of the respondents answered that their processes make out of date documents impossible. 2% would compare it to the code or existing features. Other 1% of the respondents mentioned some mechanisms or tools that are able to check the validity of the specification before work starts. Rest of the responses either did not understand the question, or could not be categorized in larger groups. For example: working on prototype means documents are always outdated, have not happened yet, "by my standards", "too bad".

5 Results through the size of the company

In this section we analyse the different mindsets through the size of the company.

We experienced that bigger companies have more career options, more experienced people, better processes, better quality validations and better on the job training instead of reliance on self-study. Bigger companies use their resources more efficiently, without overloading them more, without indirectly forcing them to produce lower quality.

In all companies with less than 1000 employees we found only 1 employee with 10+ years of experience, while 1000+ employee companies employ 6%.

As the size of companies grows, more job roles appear (1-10: 5; 11-50: 7; 51-150: 7; 151-1000: 8; 1000+: 9).

The larger the company is the more developer mindset is demonstrable. In companies with 1-10 employees three times more people selected 5 (most important) for the importance of the developer mindset than 1 (least important). In 1000+ companies the ratio is twenty three. The same is true for the testing mindset with multipliers in the range of 2-150. In the case of the management mindset the multiplier is 2-3 in all company size ranges. Technical writer mindsets are the most important in 51-150 employee companies, but on absolute scale they receive the most 5-s in 1000+ employee companies (10%).

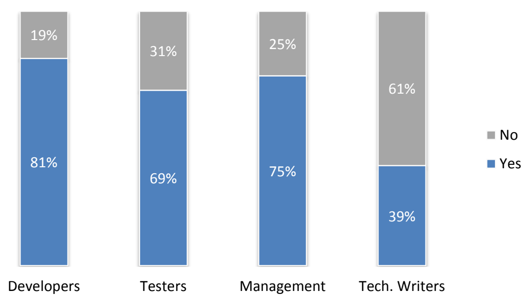


Figure 16: Manual code reviews for internally developed products.

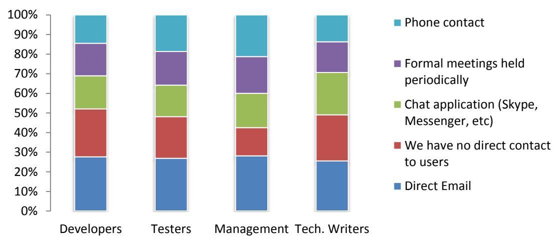


Figure 17: Types of direct contacts with customers.

We observed (Fig. 18(a)) that the average ratio of the on-the-job training gained knowledge is larger in bigger companies. While at smaller companies employees get only ∼11% of their knowledge through on-the-job training, at 1000+ companies this ratio is 38%. For self-study we observed the opposite trend: as the size of the company increases the average ratio of knowledge gained through self-study decreases from 53% to 37%. The size of the company had no effect on the average ratio of trial and error and formal training based knowledge retrieval.

Regarding the methodology related questions we found that the size of the company has a noticeable impact on the quality: almost all investigated characteristics were slightly increased/improved with the size (Fig. 18(b))).

The size of the company has a negative linear correlation with the number of people being idle in the previous year. The average idle time was 23% in the smallest companies while 12% at the largest. We found no correlation between the company size and the number of people overloaded: in companies of 151-500 employees the average overloaded time ratio was 18%, while at other companies 27-32%.

In 1000+ employee companies 24% of respondents knows and tries to remove anti-patterns and 2% uses tools for this. In all other company size ranges there were only a few respondents per size range applying tools for detecting and removing anti-patterns. The ratio of those who know and try to avoid anti-patterns is ∼33% in companies below 500 employees (17% at 501-1000 and 23% at 1000+ companies).

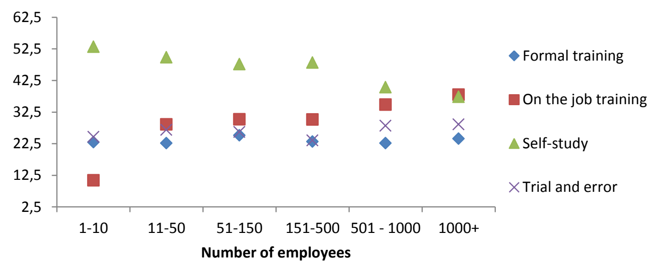
Independently of the size of the company there are two times as many respondents correcting internal quality when there is an issue than those who can correct them during development and testing without planned quality improvement sessions.

In company sizes above 150 employees ∼6-10% of employees produced low quality solutions 10+ times to satisfy short term needs. In smaller companies this ratio was 13-25%. In companies below 1000 employees only a few respondents answered producing quality without compromises. In contrast, in 1000+ employee companies it is ∼16%.

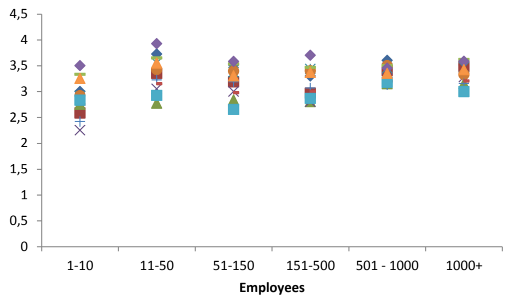
With the size of the company the existence of manual code reviews performed were growing as expected: 33% at 1-10 sized companies, ∼60% between 10-500 sized companies and 80% at 500+ companies. The duration of the manual code reviews were: 20 minutes at 1-10 sized companies, 45 minutes at 11-50 sized companies, 35 minutes at 51-150 sized companies and 65 minutes above (in average).

6 Results through experience level

There are various ways to consider experiences. Our one of the most surprising observation was that spending more time at the same working place improves the way of thinking only a little.



(a) The average ratio of knowledge gaining methods depending on the size of the company (in percentage).



(b) The average points reported for each method/thinking related question, shown by the size of the company

Figure 18: Values in relation to the size of the organization.

We were interested in where the experienced employees are working. The respondents having 10+ years of experiences consist of ∼7% of all employees, ∼14% have 6-10 years, ∼26% have 3-5 years and ∼53% of the employees have less than or equal to two years of experiences. Figure 19 shows the distribution of experiences in various team sizes.

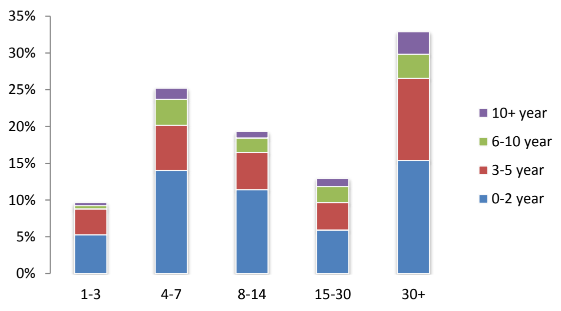
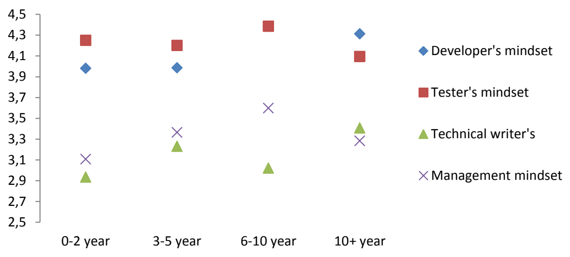
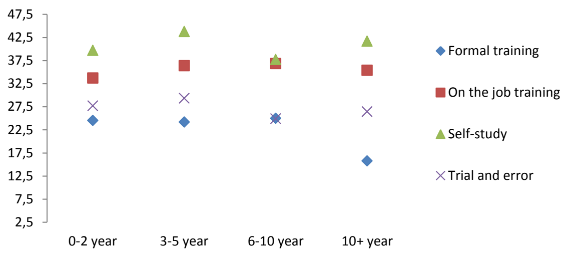


Figure 19: The distribution of experiences in various team sizes.

The importance of the mindsets is similar in all experience ranges (Fig. 20(a)). We measured that technical writer mindset gets more important with experience, the management mindset drops back in the 10+ years experience group. The developer and tester mindsets did not change significantly with the experiences.



(a) The average importance of mindsets by experience groups (in percentage).



(b) Acquiring knowledge by experience groups (in percentage).

Figure 20: Values in relation to experience.

In all experience groups the average amount of knowledge, used in work, acquired through on-the-job training/self-study/trial and error is approximately constant, while the average amount of knowledge gained through formal training drops from ∼24% to ∼16% at 10+ years of experience (Fig. 20(b)).

We observed as well that the knowledge of design patterns, testing and management techniques known does not depend on the respondents working experiences. However, technical writer techniques knowledge changes with working experience: the importance of user documentation, system documentation and reviews rises until 10 years of experience. After 10 years of experience the importance of user and system documentation no longer increases: reviews and user testing fall back. Proofreading shows an opposite trend: its usage drops back with experience, but after 10 years of experience it becomes the 3rd most known technique.

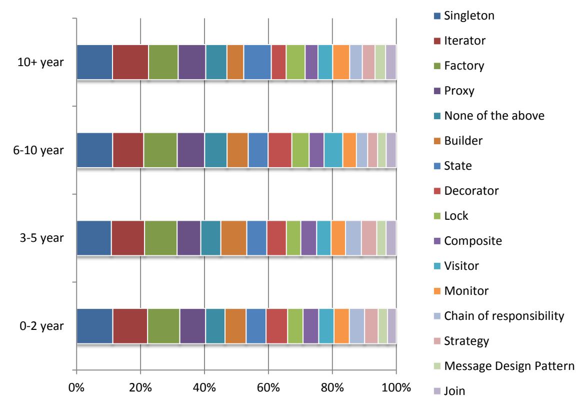
We examined the experience through the thinking/method related questions. We found that the answers for almost all questions were approximately the same in all experience groups. Until 6-10 years of experience the most improved properties were: understanding who has to carry out the next step (17% increase) and checking how the change was done when the result is not as expected (11% increase). Some properties even fall back: monitoring and evaluating the newest technologies/methodologies (18% drop), detecting if someone is idle for too long (11% drop) and learning why a non-specific activity is needed (18% drop).

The biggest progression happens between 6 and 10 years of experience: monitoring and evaluating the newest techniques/methodologies (44% increase), extensive testing before introduction (31% increase), learning why a non-specific activity is needed (30% increase).

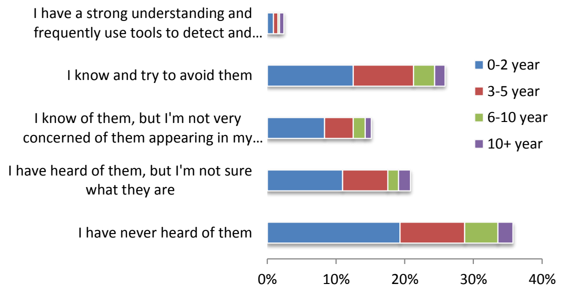
The average amount of being idle drops from 12-14% to 4% when reaching 10+ years of experience. The average amount of being overloaded slowly grows from 25% to 31%.

In all experience groups the ratio of respondents using tools to detect and remove anti-patterns was under 2%. The ratio of respondents who know about anti-patterns and try to avoid them was between 20-30% in all experience groups.

From all experience range the employees traced back the issues to the same sources with the same ratio with one exception: only 1 respondents with 10+ years of experience selected user support as the source of problems. He also placed more emphasis on reviews than others with the same or less experience.



(a) The distribution of known software design patterns by experience groups.



(b) Knowledge of design anti-patterns and experience.

Figure 21: The distribution of known software design patterns and knowledge of anti-patterns by experience groups.

After 10 years of experience all employees are working some time on internal quality improvements. The ratio of regularly improving internal quality was the highest in this group: 50%. At the same time 43% of them is improving internal quality in his free time, while only ∼30% of people with less experience reported the same.

With the amount of experience the average time spent at manual reviews rises from 40 to 75 minutes.

7 Summary

This paper reports on the findings from a survey conducted on 456 respondents working in software development projects. In order to better understand how people with different roles, experience levels or working in different size communities differ we compared: (1) how much they know and how important they perceive their roles, (2) how scientific they perceive their processes to be, (3) how they think about and handle internal quality issues.

We have seen that software is mostly developed in large companies, by people with little experience (mostly less than 2 years) in their role. According to our survey most respondents turn to the Internet, colleagues or books to learn new skills (RQ3). Formal training is shown as the least effective form of gaining new knowledge, even trial and error is perceived to be more useful on average (RQ4). For all roles surveyed on the job training and self study were reported to bring the most benefits.

Among the respondents both developer and tester mindsets are recognized to be very important in software development projects (RQ1). This result reflects, that as software systems grow, their testing also becomes a harder problem, that needs special attention. But while testing techniques are well known, development techniques rank as the least known ones (RQ2). This seems to indicate, that testing is bigger/more important, needs the participation/cooperation of more people, with diverse backgrounds.

All our questions related to scientific thinking, received answers between 3-4 points on average. There was little difference when we controlled against role (RQ5), experience level (RQ9) and size of the company (RQ8). Showing that people in different roles think alike and/or follow similar processes. The quality of thinking and processes in the case of developers and testers was the most similar: their average answers correlated with a 0.93 coefficient. Some researchers (e.g. [13], [14], [15]) have already noticed a kind of "convergence" between testing and development, our data seems to support.

With experience the usefulness of on the job training raises at the cost of self study. When estimating how scientific their thinking/processes are managers give the most spread out answers.

We found that most companies support the improvement of internal quality, but most respondents have never heard of or are not concerned about anti-patterns (RQ6, RQ7). Most of the issues are traced back to the early stages of the process and are controlled by static tools, manual code reviews and direct contact to customers.

The responses show, that bigger companies have more career options, more experienced people, better processes, better quality validations and offer better on the job training instead of relying on self-study (RQ8). Bigger companies also use their resources more efficiently, without overloading people more, without forcing people to produce lower quality.

There are two ways to think about our results regarding time. Either our results show that with spending more time at the same place the methods/thinking improves only a little bit, or we are witnessing a slow degradation of methods/thinking since 10 years ago (RQ9). This question needs to be investigated further.

The biggest difference in how people with 10+ years think better is related to how they monitor newest technologies/methods and how extensively they test them before their introduction. An interesting question can be asked: if formal training is perceived to be the least effective, people learn the most from the Internet and colleagues during on-the-job training and self-study, the maximum of knowledge (perceived) can be reached in 2 years ... should large companies still expect university degrees or rather recruit people without degrees and train them in-house?

8 Threats to validity

This study might suffer from the usual threats to external validity. There might be limits to generalizing our results beyond our settings (we only contacted companies with offices in Hungary).

The questions we have selected to measure how well the various techniques of the different roles are known, cover only a small part of the fields knowledge, and these techniques might not be used on a daily level.

We also can not claim to have eliminated cultural biases. In different countries, different domains and involving different forums we might get different results (e.g. we noticed too late that user experience related roles were left out).

When we noticed that the number of technical writers filling in our survey might not be enough, to make comparisons, we contacted the Technical Writers facebook group. As they might not be working at companies with offices in Hungary, this slightly changes the target group of our survey. We believe, that the 28 technical writers who filled in the survey, did not change the results of other groups and provided a better representation of their own group.

9 Acknowledgements

We would like to thank all those people who offered help distributing our survey. We would like to thank companies allowing their employees to fill in our survey (for example Ericsson, Nokia, LogMeIn, NSN, SAP, NNG, Prezi, GE). Our thanks goes also to the meetup groups allowing us to reach their members (Test & Tea, Hungarian C++ Community, Budapest DevOps Meetup, Freelancers in Budapest) and to all visitors of the Hungarian IT professionals group at `www.linkedin.com`, `www.hup.hu` and `www.prog.hu` who filled in our survey.

Special thanks goes to the leaders of the Technical Writers facebook group, by whom we were able to reach more technical writers.

References

[1] M. Fowler, *Refactoring: Improving the Design of Existing Code*, Addison-Wesley, 1999. ISBN: 978-0201485677

[2] E.V. Emden, L. Moonen, *Java Quality Assurance by Detecting Code Smells*, WCRE, 2002, 97-108.

[3] N. Moha, Y. Guhneuc, L. Duchien, A.L. Meur, *A Method for the Specification and Detection of Code and Design Smells*, IEEE Transactions on Software Engineering, valume 36 (issue 1), 2010, 20-36. ISSN: 0098-5589, DOI: 10.1109/TSE.2009.50

[4] H. Neukirchen, M. Bisanz, *Utilising Code Smells to Detect Quality Problems in TTCN-3 Test Suites*, Proceedings of 19th IFIP TC6/WG6.1 International Conference, TestCom 2007, 7th International Workshop, FATES 2007, 2007, 228-243. ISBN: 978-3-540-73065-1, DOI: 10.1007/978-3-540-73066-8_16

[5] J. Carr, *TDD anti-patterns*. http://blog.james-carr.org/2006/11/03/tdd-anti-patterns/, Visited: 2015.

[6] A. Scott, *Introducing the software testing ice-cream cone (anti-pattern)*. http://watirmelon.com/2012/01/31/introducing-the-software-testing-ice-cream-cone/, Visited: 2015.

[7] N. Juristo, A.M. Moreno, and S. Vegas, *A Survey on Testing Technique Empirical Studies: How Limited is our Knowledge*, In Proceedings of the 2002 International Symposium on Empirical Software Engineering (ISESE '02). IEEE Computer Society, 2002, 161-172. DOI: 10.1109/ISESE.2002.1166935

[8] A.M.J. Hass, *Guide to Advanced Software Testing*, published by Artech House, March 30, 2008, ISBN-13: 978-1596932852

[9] I. Stamelos, R. Charikleia, T. Poramen, E. Berki, *Software Project Management Anti-patterns in Students' Projects*, http://www.sis.uta.fi/t̃p54752/pub/Anti-patternsinStudentsProjects.pdf Visited 2015.

[10] W. Brown, R. Malveau, H. McCormick, T. Mowbray, *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*, Wiley Computer publishing, 1998. ISBN: 978-0-471-19713-3

[11] I. Stamelos, *Software project management anti-patterns*, 2010, Journal of Systems and Software, Elsevier, vol. 83, 52-59. DOI: 10.1016/j.jss.2009.09.016

[12] G.J. Alread, C.T. Brusaw, W.E. Oliu, *Handbook of Technical Writing*, published by Bedford/St. Martin's, 2011. ISBN-13: 978-0312679453

[13] SOASTA, http://www.soasta.com/blog/could-developers-be-the-future-of-software-testing/, Visited 2015.

[14] K. Katdare, http://www.crazyengineers.com/threads/career-in-software-testing-vs-software-development.67131/, Visited 2015.

[15] S. Rowe, http://blogs.msdn.com/b/steverowe/archive/2007/02/13/hiring-great-testers-how-important-is-testing-affinity.aspx, Visited 2015

[16] A. Yamashita and L. Moonen, *Do developers care about code smells? An exploratory survey*, 20th Working Conference on Reverse Engineering, WCRE, 2013, 242-251. DOI: 10.1109/WCRE.2013.6671299

[17] State of Testing Survey report: http://www.practitest.com/wp-content/uploads/2015/07/State_of_Testing_Survey_2015.pdf, Visited 2015.

[18] K. Szabados, *Structural Analysis of Large TTCN-3 Projects*, 2009, in proceeding of: Testing of Software and Communication Systems, 21st IFIP WG 6.1 International Conference, TESTCOM 2009 and 9th International Workshop, FATES 2009, Eindhoven, The Netherlands, November 2-4, 2009. DOI: 10.1007/978-3-642-05031-2_19

[19] ISTQB Worldwide Software Testing Practices Report 2015-2016. http://www.istqb.org/references/surveys/istqb-worldwide-software-testing-practices-report-2015-2016.html, visited 2015.

A Survey Questions

Here are the survey questions. The layout below is simplified to meet space limitations. We have noted within /**/ comments the different types of responses expected if not listed here.

A.1 Generic information

1. Are you working for a multi-national company? (A company present in several countries.) /* yes-no */
2. How large is the company you are working for? (The number of employees working in your country.)

   (a) 1-10 employees (b) 11-50 employees (c) 51-150 employees (d) 151-500 employees (e) 501 - 1000 employees (f) 1000+ employees
3. How many people are you working with in your main project?

   (a) 1-3 (b) 4-7 (c) 8-14 (d) 15-30 (e) 30+
4. For how long have you been working in your current position?

   (a) 0-2 year (b) 3-5 year (c) 6-10 year (d) 10+ year
5. What is your predominant role or responsibility within your organization?

   (a) Development (b) Testing (c) Architect (d) Technical writing (e) Team leadership (f) Project management (g) Business operation/support (h) Executive management (i) Managing of managers (j) Line management (k) Self-employed
6. What was your main task in 2014?

   (a) Requirement gathering (b) Research (c) System development (d) Writing conceptual information (e) Code editing (f) Code review (g) Deployment (h) Testing (i) Test review (j) Writing documentation (k) Maintenance (l) Managing the environment (m) Managing people (n) Administration (o) Managing Projects (p) Sales
7. What other responsibilities did you have beside your main task in 2014?

   (a) Requirement gathering (b) Research (c) System development (d) Writing conceptual information (e) Code editing (f) Code review (g) Deployment (h) Testing (i) Test review (j) Writing documentation (k) Maintenance (l) Managing the environment (m) Managing people (n) Administration (o) Managing Projects (p) Sales

A.2 Familiarity with different techniques

8. Which of the following software design patterns are you familiar with?

   (a) Builder (b) Factory (c) Singleton (d) Decorator (e) Composite (f) Proxy (g) Iterator (h) Chain of responsibility (i) State (j) Visitor (k) Strategy (l) Join (m) Lock (n) Message Design Pattern (o) Monitor (p) None of the above
9. Which of the following testing techniques are you familiar with?

   (a) Function testing (b) Boundary value anaysis (c) Decision table testing (d) Pairwise testing (e) Classification tree method (f) Statement testing (g) Branch testing (h) Exploratory testing (i) Fault attack with defect checklist (j) Error guessing (k) Cause-effect graph (l) Use-case testing (m) Path testing (n) Fault injection (o) Control flow analysis (p) Coding standard (q) Code metrics (r) Call graphs (s) Review (t) Walk-through (u) Inspection (v) None of the above
10. Which of the following techniques/methodologies are you familiar with?

    (a) Sequential development (b) Waterfall (c) V-model (d) Spiral model (e) Extreme programming (f) Scrum (g) Kanban (h) Agile (i) Test Driven Development (j) Feature Driven Development (k) Acceptance Test Driven Development (l) Continuous Integration (m) Integration Centric Engineering (n) Lean Development (o) 6 Sigma (p) Pair programming (q) CMMI (r) Planning poker (s) Refactoring (t) None of the above

11. Which of the following technical writing techniques are you familiar with?

    (a) Analysis of audience (b) Gathering specific vocabulary (c) Precise expressions (d) Clear design (e) Chain of new concepts (f) Review (g) i18n (h) L10n (i) Survey (j) User documentation (k) System documentation (l) Documentation Life Cycle (m) Problem-Method-Solution (n) Chronological structure (o) User testing (p) Camera-ready (q) S-V-O structure (r) Proofreading (s) Interview (t) Focus groups (u) None of the above

12. In your opinion how important is to have a a developer's mindset for your work? /* marks between 1 and 5 */

13. In your opinion how important is to have a tester's mindset for your work? /* marks between 1 and 5 */

14. In your opinion how important is to have a technical writer's mindset for your work? /* marks between 1 and 5 */

15. In your opinion how important is to have a management mindset for your work? /* marks between 1 and 5 */

A.3 Gaining new knowledge

16. What are your main sources of gaining new knowledge?

    (a) Books (b) Research papers (c) Colleagues (d) Classes (e) Trainings (f) Vendor sites (g) Internet forums and blogs (h) Company intranet (i) Conferences (j) Other;

17. Which of the following resources did you use to learn last year?

    (a) Books (b) Research papers (c) Colleagues (d) Classes (e) Trainings (f) Vendor sites (g) Internet forums and blogs (h) Company intranet (i) Conferences (j) Other;

18. How much of the knowledge you need in your work have you acquired through formal training? (Percentage between 0 and 100)

19. How much of the knowledge you need in your work have you acquired through job training? (Percentage between 0 and 100)

20. How much of the knowledge you need in your work have you acquired through self-study? (Percentage between 0 and 100)

21. How much of the knowledge you need in your work have you acquired through trial and error? (Percentage between 0 and 100)

A.4 Process and methodology related questions

22. In our company we are monitoring and evaluating the newest technologies/methodologies. /* marks between 1 and 5 */

23. When a new piece of technology/methodology is available we do extensive testing before introducing it into our processes. /* marks between 1 and 5 */

24. When a new activity/artifact is defined we establish sets of hypotheses that can be tested before work starts. /* marks between 1 and 5 */

25. When an activity is not done as specified we follow a defined process to improve. /* marks between 1 and 5 */

26. When we see a defective outcome despite all activity done as specified, we modify the processes. /* marks between 1 and 5 */

27. In 2014 in my opinion I was idle ...% of my time: /* asking for percentage */

28. As far as I can tell, when someone is idle for long, our team is able to detect the situation and follow a defined process to modify or reassign activities. /* marks between 1 and 5 */

29. In 2014 in my opinion I was overloaded ...% of my time: /* asking for percentage */

30. As far as I can tell, when someone is overloaded for long, our team is able to detect the situation and follow a defined process to modify or reassign activities. /* marks between 1 and 5 points */

31. If we find that a non-specific activity is needed, we learn why it is needed and redesign our processes. /* marks between 1 and 5 points */

32. In most cases in the processes we follow I find it clear what the next activity is . /* marks between 1 and 5 points */

33. I find it clear who has to carry out the next activity in the processes we follow. /* marks between 1 and 5 points */

34. When we plan to make a change we asses the current state of affair with scientific rigor. /* marks between 1 and 5 points */

35. When the result of a change is different from the expected we check how the change was done, what effects it had and redesign the change if needed. /* marks between 1 and 5 points */

A.5 Anti-patterns

36. How familiar are you with design anti-patterns?

    (a) I have never heard of them (b) I have heard of them, but I'm not sure what they are (c) I know of them, but I'm not very concerned of them appearing in my work (d) I know and try to avoid them (e) I have a strong understanding and frequently use tools to detect and remove anti-patterns

37. How concerned are you about the presence of anti-patterns in your products? /* marks between 1 and 5 points */

38. How often do you work on existing products to improve their internal quality without changing their external behaviour?

    (a) Never (b) Seldom (c) Sometimes (d) When absolutely necessary (e) On a regular basis (f) Such work is planned and done a formal activity.

39. Is working on existing products to improve their internal quality supported by your organization? (Only internal quality, without changing external behaviour)

    (a) No (b) In theory (c) Tools are available (d) When we have free time (e) We have allocated time for this kind of work in our processes

40. If internal quality improvement is done, when is it done?

    (a) We don't perform internal quality improvements (b) When there are issues, we correct it (c) When we notice a possibility to improve we take it immediately (d) We have pre-planned sessions and work lists for internal quality improvements

41. During 2014, how many times did you have to produce solutions you felt they were of low quality in order to satisfy short term needs?

    (a) Never (b) 1-2 times (c) 3-5 times (d) 6-10 times (e) 10+ times

A.6 Static analysis and traceability

42. Which tool supported static analysis techniques are used in your organization?

    (a) Checking of static metrics (b) Checking of coding standards (c) Control flow analysis (d) Data flow analysis (e) Other tools supporting static analysis (f) Our techniques are not tool supported

43. Do you have manual code reviews for internally developed products? /* yes - no question */

44. How long does a manual review take? (In minutes): /* expecting a number */

45. In your opinion, which stage could be the issues found in the last year traced back to?

    (a) Requirement collection (b) Concept/System design (c) Code writing (d) Documentation (e) Review (f) User support (g) Management of tasks (h) Management of people

46. How do you judge if a specification is out-of-date? /* free text expected */

47. What kind of direct contact do you have with your users?

    (a) Phone contact (b) Chat application (Skype, Messenger, etc.) (c) Direct Email (d) Formal meetings held periodically (e) We have no direct contact to users (f) Other: /* free text expected */