Software Development Practices in New Zealand

Diana Kirk and Ewan Tempero

Department of Computer Science

June 2012

Executive Summary

This report presents the results of an on-line survey involving 195 employees from 51 New Zealand software organisations. The survey was open for participation between July 2010 and April 2011. Research project duration was 20 months, representing approximately 1200 person-hours effort. The aim for the survey was to understand what practices are used by New Zealand (NZ) software organisations to develop software. The overall goal of our research is to increase our understanding of the mechanisms underlying current practice selection in order that we might advise the industry in a meaningful way and based on 'as-is' rather than 'should-be'.

The received wisdom for developing software-intensive products is that software organisations should choose a particular development methodology and must adhere to this methodology exactly to ensure all aspects of the development lifecycle are properly covered i.e. there are no gaps. However, we have observed that many organisations do not adhere to specific process models and this raises the question of whether the practices implemented in an organisation form a coherent set. For example, do implemented practices address quality in the delivered software, or is there a 'gap' which might lead to customer dissatisfaction?

Our results indicate that organisations do not follow standard process models such as Waterfall, SCRUM or XP but rather have established a set of practices that are suited to the particular contexts of the organisation. Furthermore, the data suggests that individuals do not follow practices in a consistent way. As exact adherence to a process is encouraged to balance performance trade-offs, the risk of gaps is clearly large in the New Zealand context. Our aims are to help organisations increase productivity and software quality by exposing gaps in their selected practices and by providing advice on how to address gaps to ensure a coherent set of practices.

Results also indicate the following:

- Software development in New Zealand tends to be implementation-centric, with a culture of collaboration, informality and reliance upon personal capability.
- Individuals tend to be involved in several aspects of product creation. This may indicate a strength in that decision-making is often carried out in a collaborative way with several roles included.
- A majority of respondents claimed to be 'agile'. This claim was backed up by extensive use of iteration, but close informal contact with customers is not practiced.
- We found major issues with clarity and accessibility of requirements. This indicates an ineffectiveness in requirements practices and represents a possible gap endemic in the NZ approach to developing software.
- We also found a significant number of issues with practices relating to product quality. These include ineffective or missing design-and-code-checking practices (for example, reviews and unit tests), a lack of independent testing of releases and patches and insufficient separation of development/test/deployment environments.
- Tools appear to be widely used in all aspects of development but generally do not drive process. The implication is that NZ organisations apply tools in an appropriate way i.e. to support the process rather than creating process around tool.
- By far the most-commented on category (both positive and negative comments) was *Process*. Practitioners in NZ believe that the processes in place in their organisations matter.

Contents

1 Introduction

The purpose of this report is to present the data submitted by participants in a recent survey of software development practices in New Zealand (NZ). The survey was open for participation between July 2010 and April 2011. The survey is part of the Software Process and Product Improvement (SPPI) project, funded by the New Zealand Ministry of Science and Innovation (previously the Foundation for Science and Technology Research - FRST).

The objective of the SPPI research is to develop and apply a range of software productivity techniques and tools to enhance the performance of the New Zealand software industry. The research has three key theme areas, relating to process and project management improvement, model driven engineering and software visualisation. Our research addresses the first of these. The aim of the survey was to understand what practices are currently used by New Zealand (NZ) software organisations to develop software. Survey findings will contribute towards our understanding of the considerations underlying current practice selection and will also help expose 'practice gaps'. These are important aspects of the development process that are missing or inadequately addressed in an organisation, leading to reduced efficiency and effectiveness and subsequent failure to meet organisational objectives, such as quality.

The long term goal for our research is to advise the industry on practices selection in a meaningful way and based on 'as-is' rather than 'should-be'. This represents an acknowledgement that many NZ software organisations do not adhere to a specific development methodology but rather have established a set of practices that are suited to the particular contexts of the organisation. Our aims are to help organisations increase productivity and software quality by exposing gaps in their selected practices and by providing advice on how to address gaps to ensure a coherent set of practices. This approach supports an improvement in the way things are currently done, in contrast to trying to force on organisations a specific 'one-size-fits-all' methodology.

Data resulting from the survey is presented below. We have included some interpretation of the data, but the main goal of the report is to enable organisations to see the submitted data and thus to gain a representative picture of software development in NZ. We hope the report will enable you to compare your organisation with the 'average' and encourage you to consider applying some practices not currently implemented by you.

In Section 2, we overview the research model upon which the survey was based. In Section 3, we introduce the study, and in Sections 4 and 5, we provide descriptive statistics for our results. In Section 6, we summarise participants' viewpoints on which practices are beneficial and which are not. We present our main findings in Section 7.

2 Theoretical framework

In this section, we overview the *software practices framework* used in our study. The framework has been published in a well-respected peer-review journal [1], where it is described fully.

The framework is based on the observation that we can understand an organisation's practices only in the context of its key objectives and operational contexts. There are many possible such objectives, for example, relating to quality, functionality, stakeholder and user satisfaction and cost. We illustrate with the objective *quality*. In Figure 1, we show the basic research framework structure with the objective *quality* along the top. The objectives make up the first framework dimension, with each objective-of-interest represented by a column in the framework.

	Quality
Define	Roadmap
	Scope
Make	Design
	Implement
	Integrate
Deliver	Release
	Support

Figure 1: Basic research framework

	Quality	
Define	Roadmap	
	Scope	
Make	Design	Template-based design document
		Design inspection
	Implement	Automated unit tests
		Unit tests reviewed by senior
	Integrate	Compliance with build process
		Dedicated test team
		Requirements based test plans
Deliver	Release	
	Support	

Figure 2: Make function quality : example 1

The second framework dimension allows us to organise practices according to *what an organisation needs to achieve* at a high level. The rationale is that focussing on the *what* (the problem) allows us to capture many different approaches to *how* (the solution) without introducing the risk of making assumptions about the solution space. For any kind of product, the producing organisation must

- Define what is to be made.
- Make it.
- Deliver it.

In Figure 1, these functional categories are shown down the left-hand side. The categories are extended to include what we believe are the main sub-categories for software. Functional

	Quality	
Define	Roadmap	
	Scope	
Make	Design	Architect is subject area expert
	Implement	Ad-hoc unit tests
		Architect available to discuss
	Integrate	Developer builds
		Client acceptance tests
Deliver	Release	
	Support	

Figure 3: Make function quality : example 2

sub-categories for *Define* relate to approaches to long-term strategic (*Roadmapping*) and short-term project (*Scoping*) product planning. Categories for *Make* consist of the generally accepted elements of making software i.e. choosing a suitable solution (*Design*), implementing the pieces of the solution (*Implement*) and putting it all together (*Integrate*). For *Deliver*, we include aspects relating to the interface between development and client organisations, namely transfer of the product (*Release*) and the product as part of the target system (*Support*). The framework does not impose any ordering of the functions. The intent of the framework is to provide a structure for asking questions and organising data, with no assumptions about practice implementations. For example, with *Make*, there is no expectation that practices associated with *Design* and *Implement* need be separate.

Each cell of the framework is a repository for information about the practices that relate to the function (row) and objective (column) for that cell. The practices in a cell effectively form the 'solution' for that cell i.e. how the organisation implements the function for the row to achieve the objective for the column.

To illustrate, in Figures 2 and 3, we show two possible approaches to achieving *quality* in the *Make* function. The two figures represent different approaches to quality. Example 1 (Figure 2) indicates a formal approach, with document templates and a dedicated test team. Example 2 (Figure 3) depicts a more informal approach, with reliance on individual capability and communications. Each may be appropriate in different circumstances. The framework permits inclusion of *any* activity that contributes towards objectives. This means, for example, that casual discussions in the lunch room may be included if practitioners believe these help with understanding required product quality.

The cells of the framework are more complex than described here, but one relevant feature is that practices in a cell are structured as practices relating to culture and those relating to techniques. Cultural aspects relate to approaches to sharing information, for example, formal documents, wiki pages, formal and informal meetings and client interfaces. Examples of techniques are pair programming, reviews, formal gates and dedicated teams.

3 Study overview

As the aim was to explore practices by asking as many practitioners as possible, we implemented an exploratory survey. We sourced candidates from on-line directories and organisations we believed may have IT groups (for example, banks, insurance groups, utilities companies and educational establishments). We also emailed government departments, user groups and software bodies with the request that our invitation be forwarded to any group the recipient believed might develop software. As a result, invitations to participate were issued to managers in 518 New Zealand organisations. Of these, 13 were returned, a followup revealed that 16 no longer exist, 30 were not involved in developing software and 12 were sole developers. Of the remaining 447 candidate software organisations, 330 did not respond, 35 declined to participate and 82 agreed to participate. During the survey period, 10 organisations withdrew as a result of time constraints and a further 21 did not participate, leaving 51 organisations participating. In accordance with ethics requirements, we asked the manager for each participating organisation to invite employees to participate, with the understanding that participation was voluntary and supplied data confidential.The 51 organisations were represented by 195 employee responses.

In Figure 4, we summarise some of the organisational information received from the manager responses. Totals do not always add up to 51 as a result of missing or invalid data.

Figure 4a shows the application areas for participating organisations. Our aim is to understand if our results are specific to a small number of subject areas only or are based on a more general coverage. With confidentiality in mind, we have grouped organisations in a way that we hope is meaningful but does not risk identification. 'Various' includes real estate, entertainment and education. We note a reasonable spread of application areas.

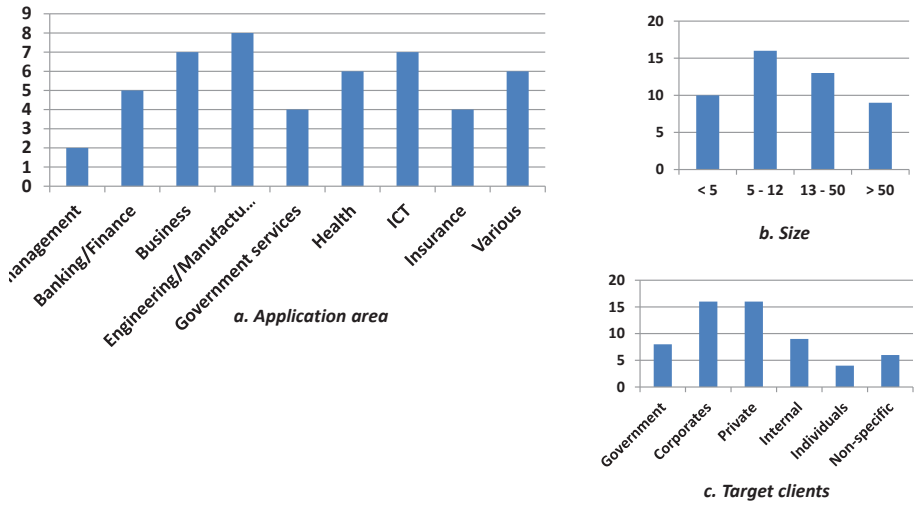

Figure 4: Characteristics of participating organisations

As we want to have some insight into possible differences between larger and smaller organisations, for example, culture and process formality, we show the spread of size in Figure 4b. Note that this metric relates to the number of people involved in the development function. In some cases, this refers to the whole organisation. In other cases (for example, an IT group within a larger organisation) it refers to the smaller group only.

Figure 4c provides a summary of the target clients. 'Government' includes entities such as district health boards, defense and police; 'Corporates' includes larger businesses and multinational corporate entities; 'Private' refers to private enterprises such as SMEs, medical practices and insurance brokers; 'Individuals' refers to individual citizens, for example, members of a club or medical patients; 'NotForProfit' includes charities and galleries; 'Internal' indicates delivery to within the larger organisation; 'Non-specific' indicates delivery to any of the above, for example, a general ICT service.

Manager participants were asked to encourage participation from within their group, according to time constraints. Approximately 40 % of organisations were represented by 3 or more employee responses, with 21 % represented by 7 or more.

In addition to a section on employee characteristics, the questionaire included sections on practices relating to culture and practices relating to techniques, grouped according to the functional categories shown in Figure 1 above. A final section asked respondents to describe the three practices that most supported their work and three that were most detrimental.

Our approach to analysis was to first analyse the complete set of responses with no regard as to contributions from different organisations. The second step was to understand how the data from individual organisations affected the overall findings. For example, a finding in the sample that a practice is 'Sometimes' carried out may be a consequence of some organisations 'Always' implementing the practice and others 'Never' implementing it. The majority of participating organisations (37) were represented by three or fewer participants and 6 by more than ten participants. We used a chi-square approach on the 6 most highly represented organisations to test for statistically-significant differences between the organisation and the sample. Although we did find some occurrences of difference, these varied from organisation to organisation and so there is low likelihood that removing two or three organisations would significantly change the distributions found in the complete sample data. This means that we will treat the descriptions presented below as characterising the complete sample set.

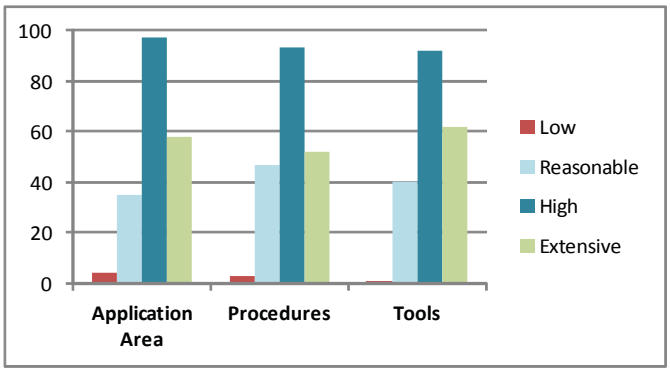

Figure 5: Employee expertise levels

4 Employee characteristics

In this Section, we present the results of analysing the characteristics of the employees respondents.

4.1 Employee expertise

We wanted to know to what extent participants were supported in their work by their knowledge and experience. We asked about knowledge of the application area, and experience with procedures and tools used. The results are shown in Figure 5. The median and mode values lie in the 'High' category, with few reports of 'Low' expertise.

4.2 Employee roles and functions

We asked participants to tell us about their roles and the functions they implement. Our aim was to understand whether organisations tended to have dedicated experts or whether employees had to be flexible and participate in a number of roles and functions. We show the results in Figure 6.

Of the 195 employees, only 64 (33 %) have 1 or 2 roles, 48 (25 %) have 3 roles and 82 (42 %) have 4 or more roles. The average number of roles per employee is 3.5 (Figure 6a). The average number of functions per employee is 5.64, with the mode at 4 and the median at 5 (Figure 6c).

The above suggests that individuals tend to be generalists rather than having specific areas of technical expertise. We next consider the possibility that the questions we asked about roles and functions may have been misleading. For example, an employee whose main job is low level design and code is likely to have also reported involvement in testing (unit testing), building (private copy before build submission) and documentation (code commenting). This may mean that the spread is exaggerated by a large number of 'coder' employees.

With all responses that included 'Coding' removed, we are left with 79 responses (Figure 7). From the figure, we see that of these responses most employees participate in between 2 and 5 functions with an average of 3.96 i.e. the spread remains substantial. We also observe from the data that most functions are well-represented.

We finally wanted to test if individuals tended to carry out specific kinds of functions. For example, are people who do product specification also involved in implementation? Do those who make the product also support it? Based on the theoretical model for this study, we divided the functions into 3 groups:

- 'S'pecification includes the problem specification functions i.e. Roadmap and Release Scope

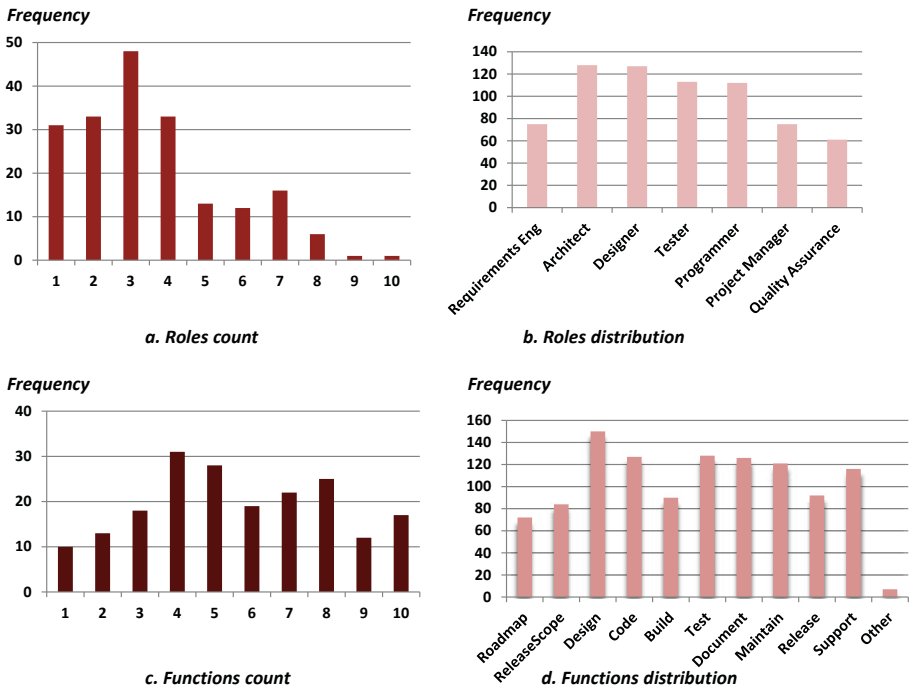

Figure 6: Number and spread of roles and functions carried out by individuals

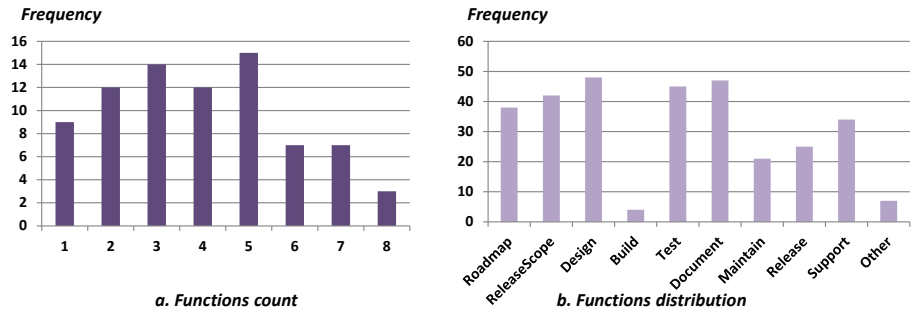

Figure 7: Function count and spread with 'Coding' responses removed

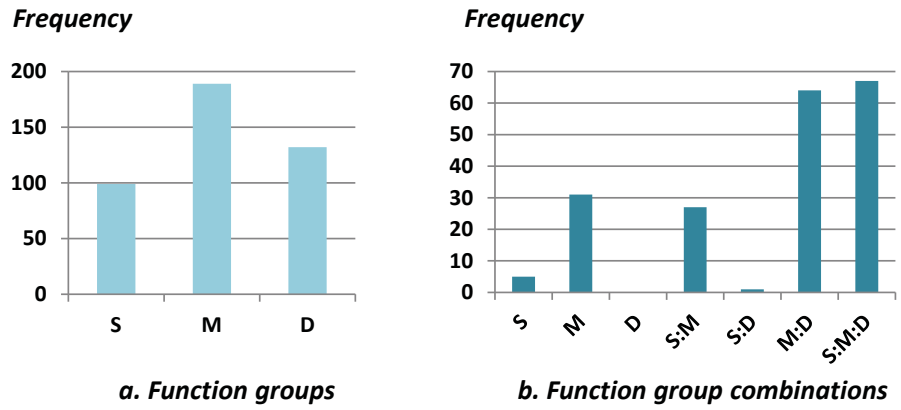

Figure 8: Function groups and group combinations

- 'M'ake includes the solution implementation functions Design, Code, Build, Test, Document, Maintain
- 'D'eliver includes the delivery functions at the client interface i.e. Release and Support

We show the results in Figure 8. From 195 employees, 67 (34 %) are involved in all groups, 189 (97 %) are involved in product implementation (Make) and 0 employees are involved in delivery functions only.

5 Organisational practices

In this Section, we overview the analysis of the practices data. In Section 5.1, we cover culture-related practices and in Section 5.2, we cover approaches to the 'Define', 'Make' and 'Support' functions.

5.1 Culture

We wanted to explore cultural aspects i.e. how organisations and individuals learn about and understand the software product(s) to be implemented and their approach to carrying out the work. We wanted to know how consistently a practice was implemented. For example, if developers report carrying out *unit testing*, do they do so in every case, or or only some of the time. We also wanted to analyse in a binary way (i.e. split responses into 'Yes' or 'No'). We provided a 4 point scale i.e. options were 'Never', 'Seldom', 'Frequently' and 'Always'.

5.1.1 Understanding the product

In Figure 9, we show the questions on how practitioners understand the product to be made. Figure 10 shows the responses for these questions. An observation is that there appears to be a lack of consistency in an individual's approach i.e. few report 'Always' or 'Never' carrying out a practice.

Question: We would like to understand how you get to know about the product you are defining, implementing or supporting. For each statement below, choose `Always' if the statement is true more than 90 percent of the time, `Frequently' if it is true 50 - 90 percent of the time, `Seldom' if it true less than 50 percent of the time and `Never' if the statement is not true.

- I consult informal documentation.
- I learn about the product at planned meetings.
- I learn about the product by informal discussions with other members of my organisation.
- I consult comprehensive documentation.
- I learn about the product by informal discussions with clients.
- I innovate based on my knowledge of the application subject area.
- I am given the information that I need.
- I have to search for the information that I need.
- I have to interpret the documentation I consult.

Figure 9: Understanding the product : questions

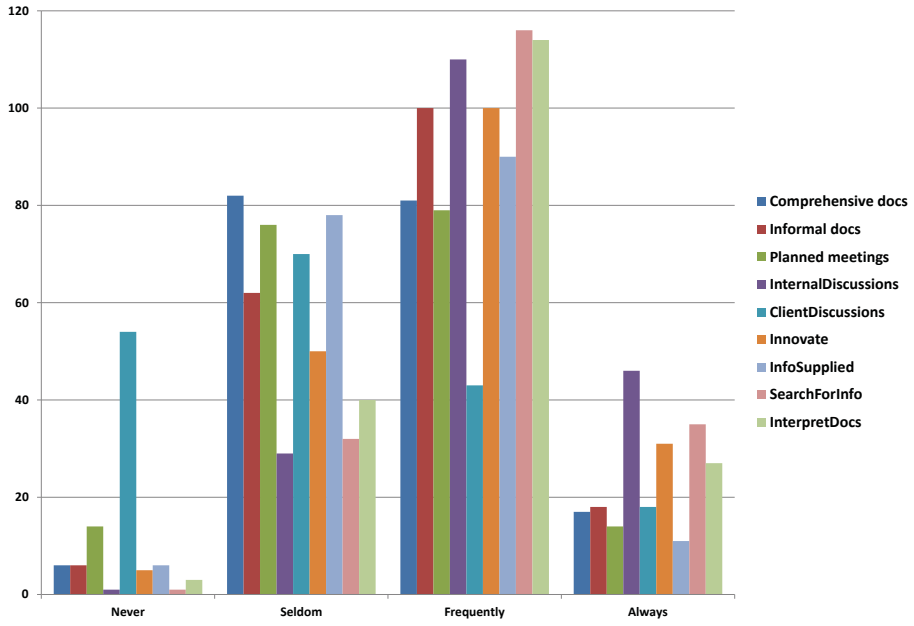

Figure 10: Understanding the product : responses

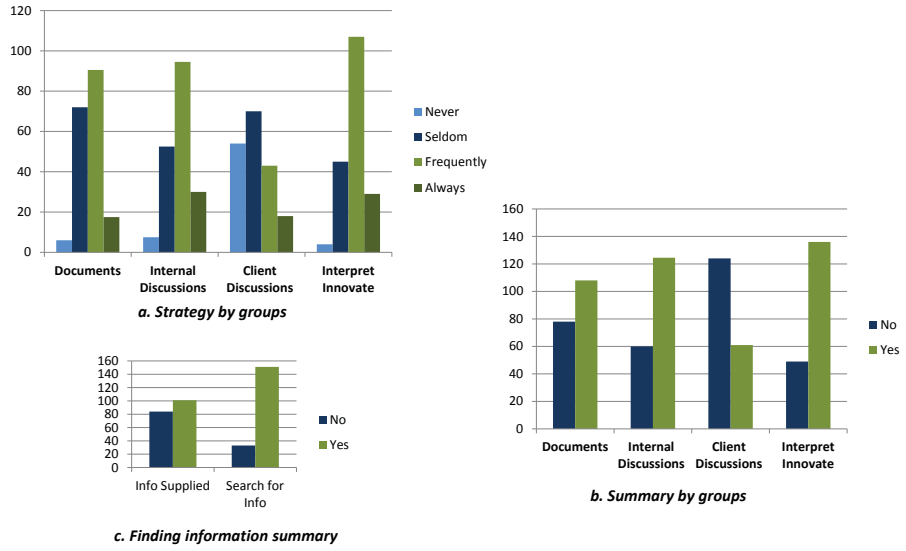

Figure 11: Understanding the product : summaries

Other observations are:

- There is an equal mix of informal and comprehensive documentation.
- Internal discussions are widely practised.
- Informal access to clients is not well supported.
- The need to innovate and interpret documentation is widespread.
- There is a need for practitioners to actively search for the information they need.

In Figure 11, we group the data in different ways. In Figure 11a, we group according to whether the source of understanding is documents, discussions within the development group, client discussions or personal interpretation and innovation. In Figure 11b, we amalgamate the 'Seldom' and 'Never' responses into a 'No' category and the 'Sometimes' and 'Always' responses into a 'Yes' category. In Figure 11c, we summarise responses relating to availability of information.

73 % of respondents report having to interpret documented instructions and/or innovate, 70 % of responses report understanding is supported by discussions with team members and only 33 % report informal discussions with clients. A culture of internal knowledge sharing and interpretation and innovation is suggested, with low access to clients. 82 % of responses report a need to actively search for the information required. 45 % of responses report that relevant information is 'Seldom' or 'Never' supplied.

5.1.2 Approach to procedures

In Figure 12, we show the questions that relate to how practitioners approach the procedures to be carried out. Figure 13 shows the responses for these questions and in Figure 14, we show the same information with responses grouped as 'Yes' ('Always' and 'Frequently') and 'No' ('Seldom' and 'Never').

Again we observe that practitioners appear to apply a range of approaches in an inconsistent way, rather than either relying on a single approach, such as following documented procedures, or consistently ('Always') implementing a practice. We also observe a reliance on collaboration and the experience and application area knowledge of individuals.

Question: We would like to understand how you do your job i.e. the kinds of procedures you apply. For each statement below, choose 'Always' if the statement is true more than 90 percent of the time, 'Frequently' if it is true 50 - 90 percent of the time, 'Seldom' if it true less than 50 percent of the time and 'Never' if the statement is not true.

- I follow documented procedures.
- I follow informally-defined procedures.
- I collaborate with others to get things done.
- I follow procedure(s) defined by the tools we use.
- I rely upon my personal experience.
- I rely upon my knowledge of the application subject area.

Figure 12: Approach to procedures : questions

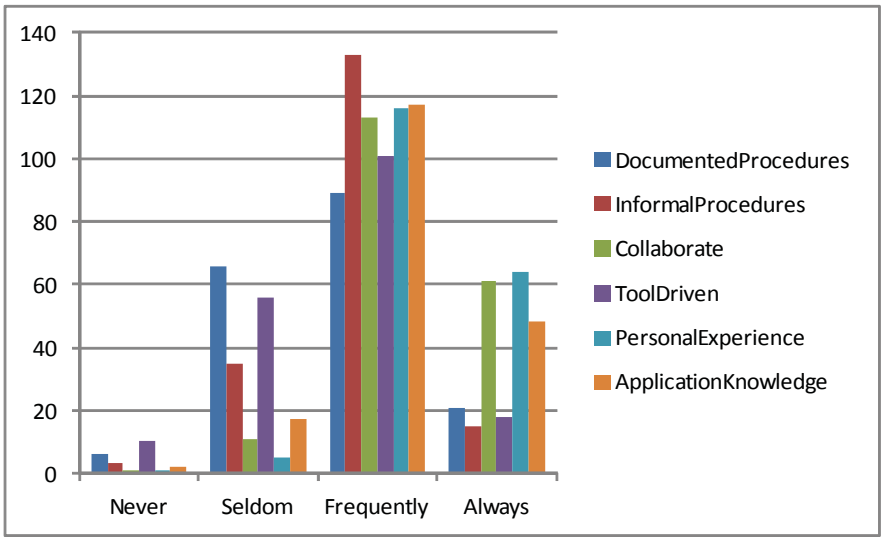

Figure 13: Approach to procedures : responses

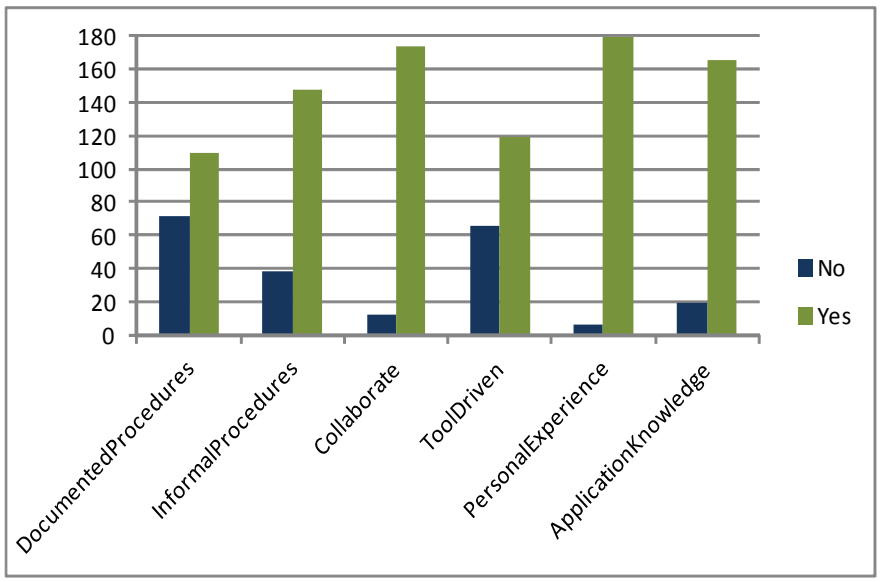

Figure 14: Approach to procedures : summary

Question: We would like to understand the practices carried out by your organisation while PLANNING your software projects. For each statement below, choose 'Always' if the statement is true more than 90 percent of the time, 'Sometimes' if it is true 10 – 90 percent of the time and 'Never' if it is true less than 10percent of the time. Please select 'Don't know' if you do not know.

- Product planning is carried out by a team.
- We produce formal planning documents.
- We receive product requirements from outside the group.
- We rely on the innovation of one or two people.
- Requirements are clearly documented.
- Details of requirements are mostly in peoples' heads.
- We plan releases well in advance.
- Everyone is kept informed about product strategy and direction.
- We formally track requirements.
- We use a tool to track features and enhancements.

Figure 15: Defining the software product : questions

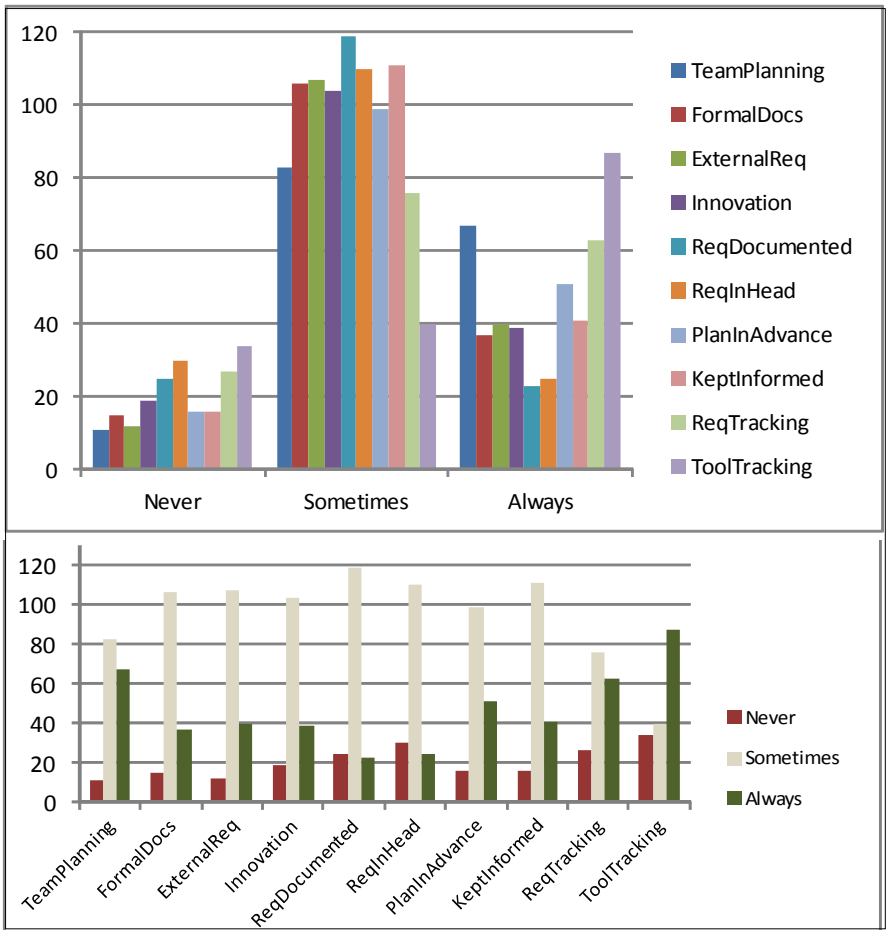

Figure 16: Defining the software product : responses

5.2 Practices

In this Section, we report responses to questions about how practitioners go about planning, implementing and delivering software products. As we would like to understand the characteristics of organisations who 'Always' or 'Never' implement specific practices i.e. focus on the extremes, we introduced a single 'middle' category and options were 'Never', 'Sometimes' and 'Always'. As a pragmatic means of highlighting 'interesting' results, we discuss responses where the 'Always' and 'Never' responses differ by a factor of 3 and the greater value represents more than 20 percent of the total responses.

5.2.1 Planning

In Figure 15, we show the questions that relate to how practitioners approach defining the product. Responses are shown in Figure 16. The two graphs represent the same information with the axes exchanged.

Some observations from these graphs and relating to the above statements are:

- 42 percent of respondents report 'always' implementing team planning for releases.
- 24 percent 'always' receive product requirements from outside the group.
- 30 percent 'always' plan releases well in advance.

5.2.2 Implementing

In Figure 17, we show the questions for how practitioners approach implementing the software product.

Responses relating to requirements, design and coding practices are shown in Figure 18. Some observations are:

- 48 percent of respondents 'always' apply an iterative approach for new development.
- 33 percent report that developers are 'always' responsible for high level design and implementation.
- 37 percent 'never' follow a strict design review process.
- 23 percent 'always' review designs informally within the team.

Responses relating to build and test practices are shown in Figure 19. Some observations are:

- 52 percent 'never' have unit tests reviewed.
- 40 percent 'never' implement test-driven development.
- 38 percent 'always' build on feature completion.
- 46 percent 'always' follow a documented build process.
- 55 percent report that developers and testers 'always' work closely together.
- 72 percent report that developers 'never' work with an allocated tester.

Question: We would like to understand the practices carried out by your organisation while IMPLEMENTING your software projects. For each statement below, choose 'Always' if the statement is true more than 90 percent of the time, 'Sometimes' if it is true 10 – 90 percent of the time and 'Never' if it is true less than 10 percent of the time. Please select 'Don't know' if you do not know.

- We apply an iterative approach for new development.
- We have dedicated people for architecture and/or high level design.
- We implement requirements traceability.
- Developers are responsible for high level design and implementation.
- We follow a strict design review process.
- We review designs informally within the team.
- Design and coding are carried out together.
- Features are formally documented and reviewed.
- We often need to be creative because features and/or enhancements are incompletely documented.
- Features and/or upgrades are incompletely documented but we have good access to clients, analysts and/or designers.
- Features and/or upgrades are incompletely documented but we have a good understanding of the application area.
- We produce unit tests for code.

- Unit tests are formally reviewed.
- We implement test-driven development.
- We build the product at least once per day.
- We build when a feature or features are completed.
- Our build process is documented and followed.
- We have a dedicated build person.
- Unit tests are run after each build.
- Code is reviewed before submitting to the build.
- We have a separate team for testing the product.
- Developers and testers work closely together.
- The developer tests the product before release.
- A tester is allocated to each developer.
- A manager, client advocate or support person helps test the product.
- All new features are independently tested by a test team.

Figure 17: Implementing the software : questions

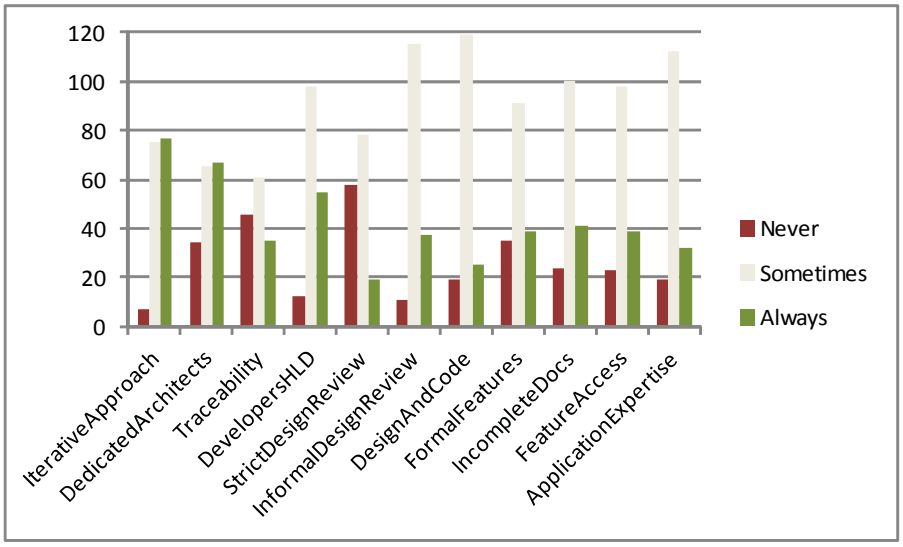

Figure 18: Implementing the software : requirements, design and code

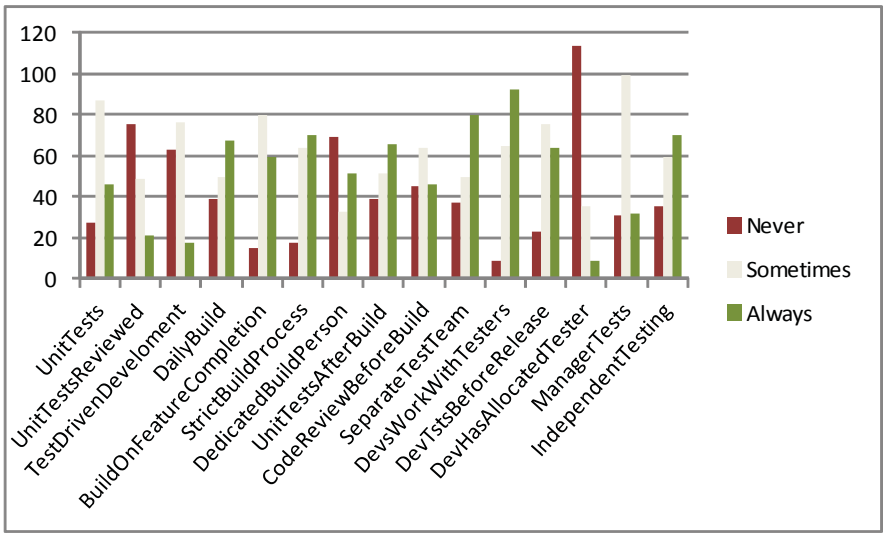

Figure 19: Implementing the software : build and test

5.2.3 Delivering

In Figure 20, we show the questions for how practitioners approach delivering the software product.

Responses relating to releasing the software are shown in Figure 21. Some observations are:

- 45 percent 'always ' have a formal release gate.
- 45-51 percent report representatives from development and QA are 'always' involved in the release decision-making.
- 73 percent report QA 'never' makes the release decision alone.
- 46 to 54 percent report 'always' releasing only fully tested new features, upgrades and patches.
- 26 percent report 'never' releasing urgent patches without comprehensive testing.
- 46 percent report 'always' collaborating closely with clients when releasing software.
- 64 percent report 'always' having a separate release environment.

Responses relating to supporting the software in the client environment are shown in Figure 22. Some observations are:

- 53 percent of respondents report that development, QA and support 'always' work closely together.
- 74 percent 'always' use a tool for tracking support issues.
- 34 percent report developers 'never' work with clients on-site.
- 80 percent 'never' having problems with on-site code changes.

Question: We would like to understand the practices carried out by your organisation while DELIVERING and SUPPORTING your software projects. For each statement below, choose 'Always' if the statement is true more than 90 percent of the time, 'Sometimes' if it is true 10 – 90 percent of the time and 'Never' if it is true less than 10percent of the time. Please select 'Don't know' if you do not know.

- We have a formal release gate.
- Representatives from marketing are involved in release decisions.
- Representatives from development are involved in release decisions.
- Representatives from QA or test are involved in release decisions.
- Representatives from client support are involved in release decisions.
- We involve clients in the release process.
- QA alone makes the decision about product release.
- We release only fully tested patches.
- We release only fully tested new features.
- We release only fully tested upgrades.
- Patches are independently tested.
- We release urgent patches without comprehensive testing.
- All patches are released through QA.
- A developer releases patches.
- We collaborate closely with clients when we release software.
- We have a separate test and release environment.

- We have a dedicated support team.
- We provide tiered support.
- Our support team, developers and QA work closely together.
- We use a tool for client issue tracking.
- Our client issue tool is separate from our defect tracking tool.
- Our support team receive formal product training.
- A dedicated person is assigned to each major client.
- Developers work with clients on-site.
- The code can be changed by a developer on-site without going through our normal process.
- We have problems with code being changed on-site.

Figure 20: Delivering the software : questions

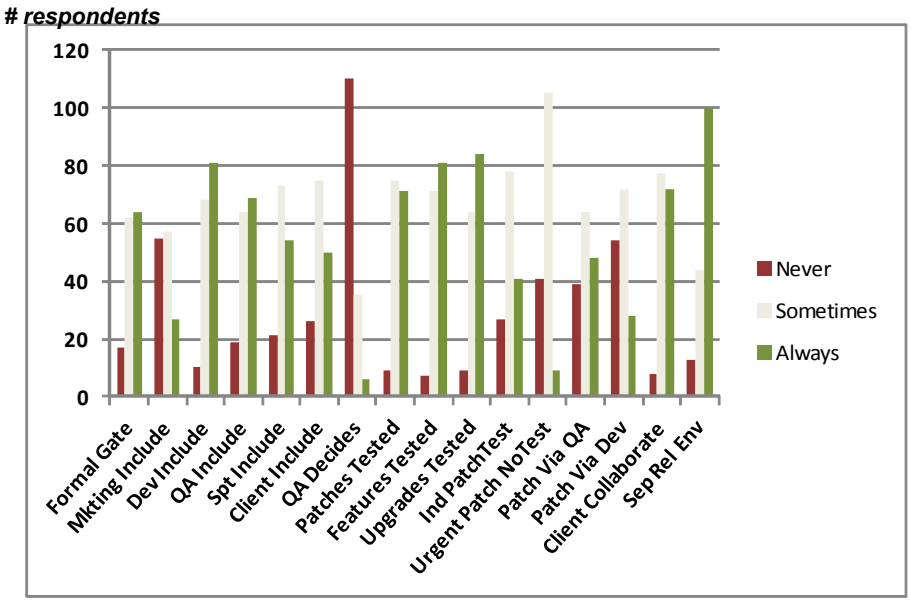

Figure 21: Delivering the software : release to client

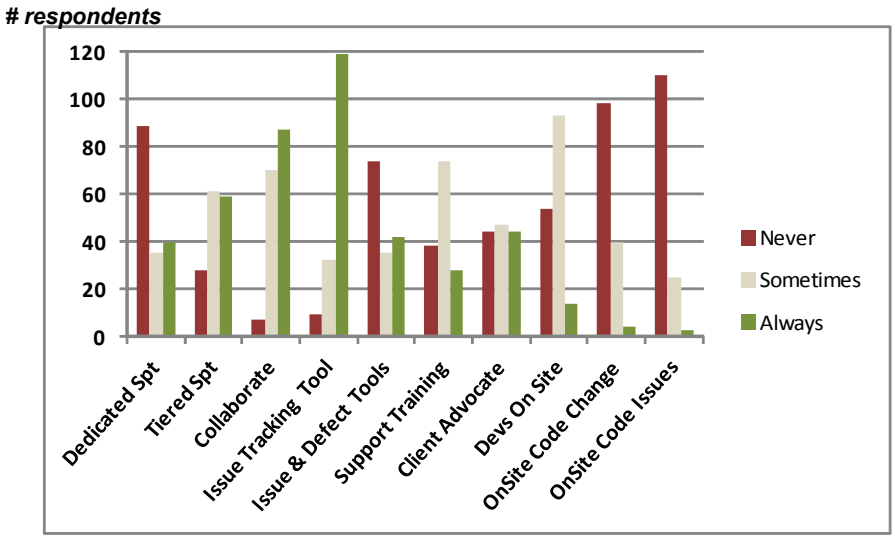

Figure 22: Delivering the software : support in client environment

6 Participant viewpoints on practices

We asked participants which practices they believed to be beneficial to the organisation and which were not. Practices other than those included in earlier questions were acceptable. In this Section, we overview the major findings. A more detailed analysis will be addressed as future work.

We grouped responses into a number of categories as suggested by the data. The categories are presented below according to the level of representation in comments:

- process
- client involvement
- team collaboration
- tools
- team structure
- documents
- team support
- individuals

6.1 Process

Participants from 41 organisations expressed opinions about approaches to process in 294 comments. 168 comments representing 36 organisations cited benefits, the most common of which were: agile, iterative approach (21 comments representing 15 organisations); approaches to managing and tracking requirements and issues (16 comments representing 13 organisations); frequent/continuous/automated builds (27 comments representing 11 organisations); release management (14 comments representing 11 organisations); design and code reviews (15 comments representing 8 organisations); unit testing (14 comments representing 8 organisations); separation of environments for development/build/test/production (9 comments representing 8 organisations).

124 comments representing 34 organisations reported unhelpful process practices, relating to: requirements and issue management and tracking (21 comments representing 18 organisations); project management (23 comments representing 14 organisations); unit testing (10

comments representing 10 organisations); design and code reviews (9 comments representing 10 organisations).

6.2 Client involvement

Participants from 23 organisations made 41 positive comments about their organisation's strong collaborative relationship with clients. Collaborations were reported at all points throughout the project and included requirements, design, testing and release. Cited benefits include: helps ensure client needs are understood; products are focussed on the needs of the client; subject area expertise can be more easily increased; problems can be more easily resolved and decisions made; clients are more enthusiastic about participation in early-adopter programmes; client numbers increase as a result of word-of-mouth advocacy.

Participants from 14 organisations reported some unhelpful client-related practices. These included poor communications, failure to set clear expectations, relying upon users to do acceptance testing, and being too reactive to individual client demands both during planning and throughout the project. Cited issues include: constantly changing roadmap; features that are unfriendly, unused or inconsistent with the planned product line; release dependencies; extensive rework to correct misunderstandings; developer frustration; unhappy clients.

6.3 Team collaboration

Participants from 21 organisations in 45 comments believe that strong team communication and collaboration contribute to success. Approaches reported included informal, open sharing of information, keeping the team informed about product direction and strategy, daily planned meetings, roles working closely together and team decision-making. Benefits cited include: reduction of gaps in outcomes as everyone knows what is going on; improved quality of ideas; identification of potential issues early on resulting in less rework and faster delivery.

Participants from 12 organisations in 15 comments believe that communications are ineffective and problematic within their organisation. Reported issues include: knowledge not shared between application area experts and developers means assumptions are made that result in later rework; development and test personnel have different understanding about the product so testers waste time testing the wrong thing; support teams not given relevant information and so developers end up supporting the support team; poor communications between developers and on-site personnel leading to ineffective fault handling; duplication of work; failure to share status information means that team members don't know about project status.

6.4 Tools

Participants from 21 organisations in 44 comments believe the use of specific tools is beneficial. Participants from 8 organisations in 10 comments believe that their organisation's use of tools could be improved.

Tools include those for project tracking, client communication tracking, requirements-enhancements-defect tracking, design, version control, client issue tracking, change requests, development environments, build and test tools and test environments.

Beneficial approaches cited include using single project tracking tool with transparent documentation so you can see project status at a glance, use of open source tools for increased development speed due to strong community support, strict work flow options so that only certain steps are possible depending upon the state the issue is in, automated build and test, source control that lets you replicate customer systems and versions, issue log linked with defect tracking tool to provide traceability and visibility for issues and all change requests in the same place.

Issues leading to ineffectiveness include: difficult-to-use or overlapping tools make it difficult to know what to do; missing, poor or out of date tool or technology in a key area; development/test environments that are difficult to set up.

6.5 Team structure

Participants from 19 organisations in 32 comments believe that the structure of the team contributes to success. Participants from 7 organisations in 11 comments express dissatisfaction with specific structures. Structures commented on included having dedicated roles for client advocacy, BA, architects and QA, and having dedicated groups for testing, customer response and release management. Most commonly cited was the need for a dedicated tester or test team (18 comments representing 12 organisations).

Cited benefits included: build product knowledge within the team (dedicated BA); build strong client relationships and trust (dedicated client advocacy); ensure designs and implementation remain focussed on the business aspect of the requirements (dedicated architect); consistency in product look-and-feel (dedicated architect); product tested from client perspective (dedicated test); client more likely to trust product changes (dedicated test); better service to clients (dedicated customer response); developers protected from client-related interruptions (dedicated customer response); better support for transition to production (dedicated release management).

Cited issues included: client is confused and does not use the appropriate support channel; developers are not sufficiently client focussed to make effective high-level design, test and release decisions; developers have increased involvement in supporting/training the support team; dependencies and bottlenecks, for example, when outsourcing development or having a single group deliver to many groups.

6.6 Documents

Participants from 16 organisations in 33 statements highlighted their organisation's approach to documentation as beneficial. Participants from 16 organisations in 22 comments believe that documentation in their organisation is unclear, piecemeal, scattered or too technical. 4 organisations in 7 comments believe that a focus on extensive documentation is detrimental.

Comprehensive and up-to-date requirements and/or technical specifications are beneficial because: result in better product understanding; need all information in one place (no one person has all the information); help clients understand what will be delivered; ensure all parties (development, QA and client) have a common understanding; provide a solid basis for testing; is vital as a starting point for understanding previous product-related decisions and existing product structure; is required as standard in the application area; ensure a consistent look-and-feel in the product or product line.

Practices reported as unhelpful include: no, unclear or scattered process documents wastes time trying to find out what to do; overly technical requirements leave the user unable to sign off; not doing enough formal design before coding so insufficient understanding; not documenting code well makes it difficult to get up to speed on an unfamiliar area; not writing test plans at the same time as requirements are being written means that features are not properly thought through; not documenting unit tests means issues aren't picked up after rework; managers talking about improving documentation but doing nothing about it.

A focus on documentation is detrimental because: adhering to existing document templates absolves the author and reviewers of any responsibility to think and so the documents end up containing little informational value and reviewers just check all the sections have been completed; tend to reuse existing templates that are inappropriate for the current project; documentation backlogs as it takes too long to upgrade documents during the project; it is not 'correct' or 'good' just because it's documented.

6.7 Team support

Participants from 3 organisations in 6 comments reported helpful approaches for supporting software teams. However, participants from 14 organisations in 19 comments reported practices that detract from the team's ability to work well.

Comments related to a proactive approach to motivating the team, to training / mentoring and to resourcing of projects.

Participants reported that insufficient training, no time to keep up to date with new technologies and the unavailability of mentors affected expertise and capability levels, leaving them unable to work effectively. Problems reported included designers unsure about architectural decisions that might impact future product direction, developers struggling to work with newer technologies, testers unsure about the details of features to be tested and support personnel being forced to interrupt developers for help in supporting clients. Participants reported issues with inappropriate resourcing strategies. These included under-resourcing of projects and being given assignments that were incompatible with personal capabilities and skills, without provision for appropriate mentoring. Also reported as problematic (7 organisations) was being called away from a development task to carry out a support task or to help the support group. While understanding the importance of the support work, participants expressed that being interrupted mid-development was detrimental to focus.

The three organisations represented in both beneficial and unhelpful practices had different aspects reported in each, for example, 'strong' in training but 'weak' in resourcing.

6.8 Individuals

Participants from 6 organisations in 10 statements commented that the levels of expertise and competence of individual team members were key to the organisation's success. Specifically commented on were specification writing by someone with very good business and product knowledge and QA test carried out by personnel with good understanding of the client business processes. Suggestions were that only extremely good people should be hired and that those who cannot abstract will always have a detrimental effect, regardless of processes implemented.

However, the risks involved in dependence upon expert staff members included loss of intellectual property on staff turnover, reliance during a project on subject matter experts who are not freely available to offer guidance and analysts who develop solutions based on instinct that is wrong and results in costly rework. Four statements representing four organisations complained about lack of project manager experience, poorly trained developers and hiring outsourced staff with low application sector knowledge.

7 Findings

In this report, we have presented the data from an exploratory survey of software practices involving 195 participants from 51 New Zealand software organisations. We have included some interpretation of the data, but the main goal of the report is to enable organisations to see the submitted data and thus to gain a representative picture of software development in NZ.

The main findings from the study are :

- Organisations do not follow standard process models such as Waterfall, SCRUM or XP, but rather adapt practices to suit the specific organisational contexts. Furthermore, the data suggests that individuals do not follow practices in a consistent way. As exact adherence to a process is encouraged to balance performance trade-offs, the risk of gaps is clearly large in the New Zealand context.

- Software development in New Zealand tends to be implementation-centric, with a culture of collaboration, informality and reliance upon personal capability.
- Individuals tend to be involved in several aspects of product creation. This may indicate a strength in that decision-making is often carried out in a collaborative way with several roles included.
- A majority of respondents claimed to be 'agile'. This claim was backed up by extensive use of iteration, but close informal contact with customers was not practiced.
- We found major issues with clarity and accessibility of requirements. This indicates an ineffectiveness in requirements practices and represents a possible gap endemic in the NZ approach to developing software.
- We also found a significant number of issues with practices relating to product quality. These include ineffective or missing design-and-code-checking practices (for example, reviews and unit tests), a lack of independent testing of releases and patches and insufficient separation of development/test/deployment environments.
- Tools appear to be widely used in all aspects of development but generally do not drive process. The implication is that NZ organisations apply tools in an appropriate way i.e. to support the process rather than creating process around tool.
- By far the most-commented on category (both positive and negative comments) was *Process*. Practitioners in NZ believe that the processes in place in their organisations matter.

Acknowledgment

This research is part of the Software Process and Product Improvement (SPPI) project, funded by the New Zealand Ministry of Science and Innovation (previously the Foundation for Science and Technology Research - FRST). We would like to thank the organisations and individuals who agreed to take part in our study.

References

[1] Diana Kirk and Ewan Tempero. A lightweight framework for describing software practices. *The Journal of Systems and Software*, 85(3):581–594, 2012. DOI: 10.1016/j.jss.2011.09.024.

Revision history

Version	Author	Description
February 2012	Diana Kirk/Ewan Tempero	First release
June 2012	Diana Kirk/Ewan Tempero	Minor updates based on analysis of contribution of individual organisations.