A Survey of Software Requirements Specification Practices in the New Zealand Software Industry

Lindsay Groves, Ray Nickson
School of Mathematical and Computing Sciences,
Victoria University of Wellington, New Zealand

and

Greg Reeve, Steve Reeves, Mark Utting
Department of Computer Science,
University of Waikato, New Zealand

May 27, 1999

Abstract. We report on the software development techniques used in the New Zealand software industry, paying particular attention to requirements gathering. We surveyed a selection of software companies with a general questionnaire and then conducted in-depth interviews with four companies. Our results show a wide variety in the kinds of companies undertaking software development, employing a wide range of software development techniques. Although our data are not sufficiently detailed to draw statistically significant conclusions, it appears that larger software development groups typically have more well-defined software development processes, spend proportionally more time on requirements gathering, and follow more rigorous testing regimes.

1 Introduction

The ISuRF (*Improving Software using Requirements Formalization*) project is aimed at demonstrating how formal specification techniques can be used to improve the requirements gathering phase of software development. This project is funded by the Foundation for Research, Science and Technology (FRST), through the Public Good Science Fund (PGSF). As the first part of the project, we have undertaken a modest survey of software development techniques used in the New Zealand software industry. Later parts of the project will involve case studies to compare formal specification techniques with the techniques currently being used, and determining what kinds of tools and techniques would facilitate the uptake of formal techniques.

The survey was primarily intended to provide a broad view of the New Zealand software industry, focusing on the techniques used for expressing software requirements and for determining whether the resulting system satisfied these requirements. We wanted to determine whether any formal specification techniques were currently being used, and what kinds of environment formal techniques would have to fit into in order to be used in practice. We also hoped this survey would be of more general interest, since we were not aware of any other such survey of the New Zealand software industry.

This report explains the method followed in conducting the survey (Section 2), presents the main findings (Sections 3 and 4), then discusses our interpretation of these findings and presents some conclusions (Section 5). More information about the ISuRF project can be found at the project web site (`http://www.cs.waikato.ac.nz/cs/Research/fm/`).

2 Method

Since this was very much an exploratory survey, we wanted to get both a broad view of the industry as a whole, and a more detailed picture of a few companies. We therefore decided to conduct the survey in two main parts: in the first part, we conducted a series of telephone interviews; in the second part, we visited four companies and conducted more detailed interviews. This section explains how we selected companies to contact, and how we conducted the two sets of interviews.

2.1 The initial contact list

Our first step was to construct a list of companies that we considered to be suitable candidates for telephone interviews. Our main criteria for selecting these companies were that they must perform software development, or develop software requirements specifications for subcontractors, and operate in New Zealand.

We compiled this list from several sources, including the "New Zealand Internet Connected Organisations " web page,[1] the Altavista search engine (searching on "software" and "develop"), the Internet Yellow Pages,[2] and a contact list compiled as part of a previous CSCW survey (see [1]). We also included a number of well-known large software companies, and a few other companies that we had personal contacts with.

In total, we had 65 companies on our contact list. We sent each of these companies a short letter introducing our project and explaining the goals of the survey, along with a two-page introduction to the idea of software requirements formalization as background for the survey (see Appendix A).

2.2 Telephone Interviews

We developed a set of questions to use as a basis for the telephone interviews (see Appendix B). The questions were designed to capture information of direct interest to our project (e.g. what software development methodologies and tools are used), plus other information that we felt might be of general interest and/or help us to identify patterns in the other data (e.g. the size of the company and the kinds of software developed).

Many of the questions were intentionally open-ended, so that each company could describe its practice in the most natural way, rather than being forced to fit their practices into a predetermined set of criteria. This inevitably leads to some difficulties in comparing the responses, and means that great care must be taken in interpreting the results.

We completed telephone interviews with 24 companies. The remaining companies were either not actively developing software or software requirements specifications, did not wish to participate, or could not be contacted. The results of the telephone survey are presented in Section 3.

2.3 In-Depth Interviews

After analyzing the telephone survey results, we selected four companies for on-site interviews, to obtain a more in-depth snapshot of software development practices in some representative companies. For this part, we restricted our attention to companies undertaking development of significant software systems, which immediately ruled out many of the companies covered by our telephone survey.

We chose four companies covering a range of sizes and applications, including one large multinational company.

We visited each of these four companies, spending about half a day at each, speaking to two or three senior staff members. The results of these in-depth interviews are summarised in Section 4.

3 Telephone Interview Results

3.1 Introduction

This section summarises the results of the telephone interviews we conducted to survey the software development techniques used in the New Zealand software industry. It contains a description of the categories in which the interview data have been presented—this includes some abbreviations for the sake of tabulation. Some of these categories (1, 4, 5 and 8) come directly from the questionnaire; the rest emerged as useful ways of summarising the responses obtained. In both cases, the categories themselves are subjective in the sense that we have imposed our own interpretation of the data

[1] See: `http://www.comp.vuw.ac.nz/~mark/netsites.html`
[2] See: `http://tdl.tols.co.nz/`

and our own ideas as to what are interesting points to bring out. This is of course usual and quite unavoidable when interpreting questionnaire results when the questions asked are of the fairly open-ended sort we have employed.

Finally, we have not summarised all of the data that we gathered—we have concentrated just on those which serve our goal of seeing how requirements gathering is done currently. The other data, of course, exist on our completed questionnaires and has served to give us a good idea of the context within which we are working and has guided us in our choice of companies to make site-visits to as well, as discussed above, as helping to shape the categories we chose for further consideration.

Presented next, in table 1, are the data in summary form and then tables 2, 3, 4 and 5 are used to suggest evidence of trends in the data.

3.2 The categories used

The interview information presented in this report is divided into eight categories, as follows:

1. *Size*.
   The number of people who are involved in software development within the organisation. This is categorized into one of the three values: small (S) representing 1-3 personnel, medium (M) representing 4-9 personnel , or large (L) representing 10 or more personnel.

2. *Kind* of development.
   The type of software development projects undertaken by the organisation. The differing projects found were grouped under the following categories:
   - Specific software products for customers.

     O : One-off contracts

     M : Mass production (Shrink wrapped)

     C : Customizing pre-developed software

     S : Service/support
   - In house development to support running of organization.

     IO : Own development

     IC : Customization of bought-in products
   - Product support. Inclusion of software in organisation's products.

     PI : In house development

     PB : Brought from outside source

3. *Formality* of process and/or notation.
   Here formality refers to both the process and the language or notation used. *Formality of process* denotes the degree to which the specification process is well-defined, documented and followed for all projects. *Formality of language or notation* refers to the well-definedness of the language or notation used to write down specifications. Specification languages with precisely defined semantics, such as Z and VDM, are regarded as fully-formal, whereas notations like UML, whose semantics are not precisely defined, are regarded as semi-formal.
   We describe the degree of formality, on a scale of 1 to 5, where 1 is the least formal and 5 is the most formal, as follows:

   1 : No explicit process and no formal language.

   2 : Clear phases, though any method used is implicit and no formal language used.

   3 : Clear phases, and a sequence of informal specifications made during a project.

   4 : Formal process, with semi-formal notation.

5 : Formal process, with fully formal notation.

This classification is very subjective because it depends on the interviewee's perception of degree of formality, what the interviewer chose to record, and our later perception of the recorded information. The phrases above suggest how we decided which category to put a company into, though the goodness-of-fit will vary.

4. *Specification* of requirements.
   Indicates an estimated proportion of the lifespan of a typical project spent on specifying requirements.
   These are mostly very rough estimates, since accurate information was often not available, especially when there was no clear requirements specification phase. Many of the companies also indicated that this varies considerably among different projects.

5. *Standards*.
   Any industry/ISO/New Zealand standards that are maintained by the organisation

6. *Testing* carried out on developed software.
   This is a rough classification of the types of testing performed into a level of rigour. An organisation is rated at a given level if they use at least one of the methods pertaining to that level (and perhaps methods from a lower level). The levels are defined as follows:

   1 : Testing by customers (after release); includes beta releases.
   Developers perform their own testing during development.
   Testing by eventual users (before release); includes integration tests on site.

   2 : Unit, system and/or acceptance testing.
   Project-specific test plans used.
   Regression testing.

   3 : Testing done by dedicated testers (not developers involved in the project).
   Test plans derived from the specification.

   This categorization is again somewhat subjective, and organisations may well use other testing methods that were not mentioned during the interview.

7. *Tools and Languages* used.
   An indication of some of the tools, languages and development methods used by the organisation.

Table 1. Data Summary

Size	Kind	Formality	Specification[a]	Standards	Testing	Some Typical Tools and Languages
L	IO,IC	3	20-40%	ISO9000	2	None mentioned
S	O	1	20%		1	C,C++,VB,Access,Code DBs,Coral Draw
M	O	2	25%		1	JBuilder,C,C++,Java,SQL
L	O	4	25%		3	UML,Virtual Modeller,VB,FoxPro,Access
L	O	3	30%		3	ERwin,Visual Source, Progress 4GL
M	O	4	40%		2	VB,Access,Delphi,C,Informix
S	O	1	15%		1	Prolog, C, ODBC, HTML, Javascript
M	O,M	4	40-60%		2	Objectory,C++,Java,SOM/DSOM(IBM),Oracle DB2,Object Store, Poet
L	O,C	4	30%	MS-Cert.	3	FoxPro,Delphi,C++,C
L	O,S	4	20%	ISO9001, CMM-2, SWIFT	3	Use tools and languages deveoped in-house and: Rational Rose, UML
M	O,IO	1	20-30%		1	DreamWeaver,Frontier,MacroMedia,html, Perl
S	M	3	*		3	Booch,Visual C++,VB,Delphi
M	M	2	15%	ISO9002	3	VB, SQL
S	M	2	20%		1	Delphi,Paradox DBs,Pascal,QuickBasic
L	M,C	4	25-40%		2	SourceSafe,UML,C,C++,Java
M	M,C	2	*		1	KLOC(EFT-pos)
M	C	2	30%	ISO9000	2	C,Ingres,VB,SQL-Server
S	IO	2	30%		1	FoxPro,RDBM
L	IO	3	35%	ISO9001	3	Oracle, SQL, C
M	IO,IC	3	*		2	Power Builder,VB,Perl, Visual C
S	IO,IC	1	10%		1	C++,Delphi,Assembler
M	IC,PB	2	10-20%		2	KBM
L	PI	3	30-40%	ISO9001	3	Templates,class diagrams,C(Unix, Win),Jade(NT),VB,C++
L	PI,PB	4	*[b]	ISO9001	3	Rational Rose C++(Booch/UML),Project Technology, Bridge Point

[a] A * in this column indicates that we were not able to ascertain the relevant percentage from the data we gathered.

[b] In this case, the company stated that the amount of specification done varied considerably from project to project—we estimate this might vary between 10% and 50%, but we show a value in the mid-point of the range in the tables that follow.

Table 2. Summary – in order of software team size

Size			Kind of Development								Formality					Specification								Testing		
S	M	L	O	M	C	S	IO	IC	PI	PB	1	2	3	4	5	10% ⟷ 55%								1	2	3
		*					*	*					*							*					*	
		*	*		*									*						*						*
		*	*			*								*				*								*
		*	*											*					*							*
		*					*						*								*					*
		*		*	*									*							*				*	
		*							*				*								*					*
		*	*										*							*						*
		*							*	*				*							*					*
	*			*	*							*												*		
	*		*				*				*								*					*		
	*		*									*							*					*		
	*						*	*					*												*	
	*							*		*		*					*								*	
	*			*								*					*									*
	*				*							*								*					*	
	*		*											*								*			*	
	*		*	*										*									*		*	
*				*									*													*
*							*					*								*				*		
*			*								*						*							*		
*						*						*						*						*		
*							*	*			*					*								*		
*			*								*							*						*		

Table 3. Summary – in order of formality

Size			Kind of Development								Formality					Specification								Testing		
S	M	L	O	M	C	S	IO	IC	PI	PB	1	2	3	4	5	10% ⟷ 55%								1	2	3
		*	*		*									*						*						*
		*	*			*								*				*								*
		*	*											*					*							*
		*		*	*									*							*				*	
		*							*	*				*							*					*
	*		*											*								*			*	
	*		*	*										*									*		*	
		*					*	*					*							*					*	
		*					*						*								*					*
		*							*				*								*					*
		*	*										*							*						*
	*						*	*					*												*	
*				*									*													*
	*			*	*							*												*		
	*		*									*							*					*		
	*							*		*		*					*								*	
	*			*								*					*									*
	*				*							*								*					*	
*							*					*								*				*		
*						*						*						*						*		
	*		*				*				*								*					*		
*			*								*							*						*		
*			*								*						*							*		
*							*	*			*					*								*		

Table 4. Summary – in order of average time spent on analysis/specification

Size			Kind of Development								Formality					Specification								Testing		
S	M	L	O	M	C	S	IO	IC	PI	PB	1	2	3	4	5	10%			⟷				55%	1	2	3
	*		*	*										*									*		*	
	*		*											*								*			*	
		*		*	*									*							*				*	
		*							*	*				*							*					*
		*					*						*								*					*
		*							*				*								*					*
		*	*		*									*						*						*
		*					*	*					*							*					*	
		*	*										*							*						*
	*				*							*								*					*	
*							*					*								*				*		
		*	*											*					*							*
	*		*									*							*					*		
	*		*				*				*								*					*		
		*	*			*								*				*								*
*						*						*						*						*		
*			*								*							*						*		
	*			*								*					*									*
	*							*		*		*					*								*	
*			*								*						*							*		
*							*	*			*					*								*		
	*						*	*					*												*	
*				*									*													*
	*			*	*							*												*		

Table 5. Summary – in order of formality of testing procedures

Size			Kind of Development								Formality					Specification								Testing		
S	M	L	O	M	C	S	IO	IC	PI	PB	1	2	3	4	5	10%			⟷				55%	1	2	3
		*	*		*									*						*						*
		*	*			*								*				*								*
		*	*											*					*							*
		*							*	*				*							*					*
		*					*						*								*					*
		*							*				*								*					*
		*	*										*							*						*
*				*									*													*
	*			*								*					*									*
		*		*	*									*							*				*	
	*		*											*								*			*	
	*		*	*										*									*		*	
		*					*	*					*							*					*	
	*						*	*					*												*	
	*							*		*		*					*								*	
	*				*							*								*					*	
	*			*	*							*												*		
	*		*									*							*					*		
*							*					*								*				*		
*						*						*						*						*		
	*		*				*				*								*					*		
*			*								*							*						*		
*			*								*						*							*		
*							*	*			*					*								*		

4 In-depth Interviews

To complement the broad-brush picture of the New Zealand software industry painted by the above survey, we visited four companies and discussed their software development practices in more detail. We chose a range of different sized companies, from medium to large. A summary of these interviews is given in this section, focusing mostly on requirements and specification issues.

4.1 Company A

Company A manufactures point-of-sale solutions, security systems and petrol pumps, for sale in New Zealand and overseas. It employs around 250 people. We focussed on the security systems division, which employs around 45 people, including 26 software developers. These developers are structured into several project groups, with each group typically comprising a manager, four to five software engineers and one to three software testers.

Company A is ISO 9001 certified, so follow a documented software lifecycle, which is generally a waterfall model, but with prototypes done during the design stage if necessary. Lotus Notes is used to track requirements, design documents, test plans and results, and processes.

A typical project begins with a *Terms of Reference* statement which is used to evaluate feasibility and guide the production of the *Functional Requirements* document. This is generally based on a standard template, and often contains *use cases*, as well as interface and performance requirements. A major requirements review meeting is held after the functional requirements are complete.

Next, work proceeds into the design and implementation phases, where design reviews, and some code reviews, are done internally by the project group. In parallel with design and coding, the testers produce test sets directly from the functional requirements document, and the design documents as they become available. They use *Cause and Effect* techniques to generate formal test plans for the product. These test plans are mostly used to perform system testing, but some modules are tested individually. This is followed by alpha and beta testing and load testing. They use some tools to help in applying tests, but this is not fully automated. They plan to start using automated regression testing tools in the future.

Typically, a third of the effort on a project is spent on feasibility and requirements gathering, a third on design and implementation, and a third on testing. They believe that it is important to get requirements agreed up-front, but it is inevitable that some clarification will come later, typically during design. They would like to do more tracking of changes, to be able to track whether anomalies were introduced during requirements, design or coding. They use the *critical chain* method to manage project schedules and are pleased with how this is working—their latest project was brought to beta test one month early.

4.2 Company B

Company B is a large New Zealand company that manufactures electronic communications products. It employs over 900 people and exports 90% of its products to over 80 countries. Its products contain an increasing amount of software, and its main development divisions each have around 30 software engineers. One of the main products produced contains embedded software which is often customised for individual customer requirements, and they maintain around 100 variants of this software. The software for another main product is more standardised, with only a few variants.

Company B is ISO 9001 certified, and is in the process of introducing a product development process which is a local variant of Hewlett-Packard's product development process. The local variant has seven phases, separated by milestones, or stage-gates (D0-D6), covering the entire product life cycle from feasibility analysis to product obsolescence.

The phase leading up to the D0 stage-gate explores the feasibility of a new product concept, including market analysis and technical feasibility. If the project proceeds, this phase will result in a product definition which still allows many possible technical solutions. The phase leading to D1 narrows this down to a single preferred "paper solution" and produces a project plan and a high-level product design. The D1 to D2 phase produces a working prototype of hardware and software, and ensures that all design risks have been eliminated. Milestone D3 marks the first production product. Between D2 and D3 the product implementation is realised, including beta testing, type approvals and customer trials.

The phase to D4 takes the product to full production, and is mainly concerned with manufacturing and assembly. The D4 to D5 phase involves on-going maintenance and upgrades. The D5 milestone marks the end of development. The D5 to D6 phase plans product obsolescence, and the D6 milestone marks the product becoming obsolete.

Software development is managed within the overall product development process. This process is quite flexible, and the project plan developed during the D0 to D1 phase determines, for each project, what will actually happen at each phase and what reviews are performed. The D0 to D1 phase produces a software specification, and perhaps some of the design. How much design is done in this phase, and the amount of time spent on capturing requirements, varies according to the project, and in particular depends on the perceived risk—for unfamiliar projects, more time is spent in this phase than with more familiar projects. A major project review of requirements design and project plan is held prior to passing the D1 stage-gate. A test plan must have been produced and executed prior to achieving the D3 (first production product) milestone. Typically the development group defines validation tests, but some larger projects have used independent testing as well and this is likely to become standard practice in the future.

The product development process also allows a variety of software development methodologies to be used, and the Company uses a mixture of "traditional" functional decomposition and object oriented methods. These are used in roughly equal proportions, with the traditional approach being preferred for hardware constrained applications.

A variety of object oriented methodologies and tools are used. There is significant use of the Shlaer-Mellor *Information Model* methodology[3] which captures designs using entity-relationship diagrams, finite state machines and data flow diagrams. Some engineers use *Rational Rose*[4] to provide tool support for software design, UML documentation and coding development, and there is some use of Booch's methodology. Because of the wide variety of products and hardware platforms, there is no standard coding language across products. Most coding is done in assembly language, C and C++, although other languages are sometimes used. Assembly language is typically used for hardware constrained systems. A variety of other tools are also used, for example MATLAB is used extensively for simulation. Some of the engineers are also conversant with SDL,[5] largely because many of the protocols used are defined using SDL.

4.3 Company C

Company C is a software development company based in Auckland, that employs approximately 20-30 staff. It specializes in leading edge development of low-level communication software. Some examples of past and present development projects include a process system for the travel industry and system integration work.

Company C uses two styles of requirements specification. The first (preferred) style is when a client comes with a requirements document, prepared by the client's own IT staff or by an independent consultant. Sometimes this requirements document includes quite detailed design information. Other times, Company C's systems analysts will develop the design, in conjunction with the client, to ensure that the solution matches the client's needs.

The second style of requirements gathering involves Company C's sales staff visiting the client and formulating requirements. This typically takes longer, because client requirements can be fuzzy. A common difficulty in the health industry is that requirements must be approved by large committees and it is difficult to please everyone. The resulting initial requirements document (containing prose and whiteboard diagrams) is then passed on to the systems analyst, who produces a high-level design (prose plus Visio diagrams and some screenshots) for approval by the client.

[3] See: `http://www.projtech.com/info/smmethod.html`
[4] See: `http://www.rational.com/products/rose/index.jtmpl`
[5] See: `http://www.sdl-forum.org/SDL/index.htm`

The time spent on requirements specification varies between projects. Company C has found that the more time spent on specification and design the more smoothly the project goes—less "reworking" is required. They are progressing toward using more detailed specifications. However, this is sometimes resisted by clients, who view work on ensuring that requirements and design are correct as being wasted time and prefer to measure progress purely by how much has been implemented.

Company C uses a variety of development methods and tools, including Rational Rose, UML and Java for some projects and Visual Design Studio and C++ for others. Integration and system testing is usually performed by the development team.

The factors that the Company perceives as barriers to spending adequate time performing requirements specification include:

– An accounting outlook towards the development process: deliverables must be made by deadline. When nothing is been produced, for example executable code, then progress is less evident.
– Customer investment: the customer has paid a proportion of the cost in deposit and wants to see (what they think is) progress. Often extensive work on ensuring the requirements are correct is not viewed as progress by the customer. However, the Company's staff believe specification and design diagrams are a good thing to be able to show customers.

Changes to the project after the point of contract are submitted to the Company in the form of a change request. The Company is very clear that changes from the contracted specification are the liability of the client, and charge accordingly.

4.4 Company D

Company D is an international company that employs approximately 300-350 staff in New Zealand. They develop software for a wide range of applications for external clients.

Company D views each of its operations world-wide as a distinct centre of expertise and encourages the incorporation of existing local knowledge and expertise into projects. While there remains some mixture of cultures between overseas and local groups, Company D is currently introducing its proprietary company-wide methodologies for developing software solutions.

The Company expects software requirements to be supplied by the client, but often finds that those requirements are ambiguous and that the client does not have a firm grasp on what they require. The Company currently spends about 20% of total development effort on the requirements and specification phase. It would like to increase this, because it has observed that the more specification that is done the easier the development stages become.

The Company made the following observations based on experience with requirements specification in New Zealand:

– There is often a large gap between the high level contractual specification produced to secure a contract and the solution specification that is needed to proceed to an implementation.
– Development of more precise contractual specifications is hindered by the competitive market. Especially when bidding for contracts—the contractual specification is often different to the solution specification.
– It would be desirable to be able to express requirements far more precisely before presenting them to the customer. This would allow for far better change tracking, ease and speed of development and completion to specification later down the track.
– A lot of the changes requested by the client are being absorbed into development because the requirements specification is not precise enough to identify these changes as deviations from the original requirements.
– Customers are often unhappy about a lot of time being spent on requirements specification and design, because it is not obvious to them that they are getting anything for their money—they expect code and some tangible product.
– Requirements gathering gets easier as knowledge of an area of expertise grows, from having done previous projects with similar solutions.

5 Conclusions

The results of our telephone survey demonstrate that there is a very wide range of kinds and sizes of organisation doing software development in New Zealand. These range from very small companies specialising in particular application areas, and small groups within companies whose main business is not software development, to large companies, or groups within even larger companies, undertaking much larger and more varied software development projects.

There is also a wide range in the kinds of applications developed, and in the kinds of client-developer relationship encountered. Some companies do only in-house development, and spend much of their effort in maintaining and upgrading a few large systems; others undertake contracts for external clients, or undertake development projects for an anticipated market (perhaps as part of another product), rather than for specific clients.

This diversity makes it difficult to identify meaningful patterns in the software development practices employed. We also need to take care in interpreting the results of our survey, due to the size of the sample, the nature of the questions, and the variability in the way answers were given and recorded. We can, however, draw a few tentative conclusions from our results.

The most significant factor in determining the kinds of practices employed appears to be the size of the software development group. It seemed, in general, that larger software development groups tend to have more well-defined software development processes. They also appear to spend proportionally more time on capturing requirements, and to have more rigorous testing regimes. It also appears that more well-defined software development processes are used by software teams that produce software for customers (either as software products or, like Company B, embedded), rather than in-house providers.

These trends may be explained by the general requirement for larger organisations to have a more formal management structure, and the fact that these organisations tend to undertake larger software development projects for which more rigorous development and testing methods are appropriate—unfortunately, our question regarding the size of system developed did not produce information allowing us to investigate that relationship.

The data concerning the proportion of time spent on capturing requirements are also difficult to interpret because many companies were not able to provide accurate data, and what data they did provide were expressed in terms of their development process. The fact that larger software development groups appear to spend more time on capturing requirements may be due to the complexity of the systems being developed, but may also be due in part to various other factors: for example, there being more client groups whose interests need to be considered, or there being less contact between developers and clients during later phases of the project. Some of the smaller companies said that they did not need to spend much time documenting requirements because they work closely with clients during the development, in some form of prototyping process. Where the developers undertake several projects for the same client, there is also a carry-over of understanding about the client's environment that allows software requirements to be expressed more concisely.

The telephone interviews and site visits also gave rise to a number of interesting comments that were not covered by the questionnaire. While these do not provide any basis for comparisons, they do present some interesting viewpoints and ranges of opinions.

Some of the people we spoke to said they would prefer to spend more time on capturing requirements, but were prevented from doing so by commercial considerations. In some cases this was because they were tendering for contracts, and had to limit their investment in the tender in case they did not win the contract—and avoid competitors gaining from them, as the tenders may subsequently become public documents. In other cases, they said that clients, having paid (some proportion of) the cost of the required system, were impatient to see what they considered to be the "product", which usually meant executable code of some sort. This attitude, clearly, militates against a process that spends time getting requirements right before moving to development.

We also encountered a wide range of opinions on the value of standards such as ISO 9001. Some people said that certification to such standards was too costly to maintain and provided no commercial advantage, while others said that these standards were very worthwhile, and that once absorbed into the corporate culture they required no additional effort to maintain. It is worth noting that many of the companies that did have ISO 9001 certification had it for parts of their operation other than software development. Some companies were planning to seek an evaluation against the Capability Maturity Model (CMM) developed by Carnegie Mellon University.[6]

In many of the companies we spoke to, the current software development process had been in place for no more than 12 to 18 months, and in several cases no projects had been completed under the current process. Some people indicated that new processes are introduced every couple of years, and others indicated that the documented processes do not really reflect what happens within their company.

Although there did not appear to be great concern about the ability of software developers to produce the software their clients require, a succession of newspaper stories would suggest those outside the industry have a different view—a recent editorial on the Incis system provides an example (see Appendix C). Whatever the factual basis of such articles, we feel the public perception is something we need, as a discipline and an industry, to be aware of. In undertaking this survey, we deliberately avoided raising questions concerning such highly publicised problems, since we felt that this might cause the representatives of these companies to become defensive and thus less open in answering other questions.

In conclusion, we wish to reiterate that this was a modest survey with modest aims, and that great care must be taken in interpreting the results. While this form of survey was quite appropriate for our purposes, to get more reliable/significant results would require a more focused survey, concentrating on a smaller range of companies, with more narrowly worded questions and more tightly defined response categories. We hope that the results presented here would provide useful background to such a study.

Acknowledgments

This report is part of the ISuRF project, supported by FRST grant UOW805.

We would like to thank the companies who participated in this survey for their time and effort.

References

1. C. Blackett and S. Reeves. CSCW in New Zealand: a snapshot. Technical Report 96/15, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 1996.

[6] See: `http://www.sei.cmu.edu/cmm/cmm.html`.

A Introductory Letter

A.1 The letter

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton
New Zealand

Telephone +64-7-838 4398
Facsimile +64-7-838 4155
Email stever@cs.waikato.ac.nz

The University of Waikato
Te Whare Wānanga o Waikato

30/10/98

«ContactPerson»
«CompanyName»
«Address»
«City»

Dear «Salutation»,

Improving Software using Requirements Formalization—ISuRF

A team of researchers based in the Computer Science Departments at the University of Waikato and Victoria University of Wellington are currently involved in the above titled research project. The research is a two year project funded by the Government's **Foundation for Research, Science and Technology (FoRST)** in a field that is of great significance to organizations and businesses that use computers. The purpose of this letter is to request your involvement in this research. From what we know of your company, we believe that software development or specification is an important part of your business.

The first phase of our research is to establish the current use or possible future use of formal approaches to software development in organizations such as yours. We want to be sure that our research is relevant to the NZ software industry, so we are conducting a survey of software developers to determine current practice, and opportunities for introducing formal techniques. We will produce a report which gives a snapshot of the potential of such methods in the New Zealand environment.

Enclosed with this letter is a short overview of the perceived potential of formal methods to software developers and specifiers. We would be grateful if you would participate in a short telephone interview to discuss your current software development techniques, the relevance of this field to your organization, and your requirements and expectations of these systems.

I encourage you to read through the enclosed material at your leisure. If you have any questions or comments, please contact me by email or phone. Someone working on the project will telephone you in August or early September to ask you the survey questions.

You can find out more about the project in the (recently established) web-page for the project at `http://www.cs.waikato.ac.nz/cs/Research/fm/`.

We realize that confidentiality is often desired in projects like this, so please be assured that any data that your organization supplies to us in the course of this survey will not be made available, in any identifiable way, beyond the members of the project team.

Your interest is greatly appreciated, and we look forward to discussing this rapidly growing field with you in person.

If you feel that you are not the best person in your organization to deal with this matter, it would be greatly appreciated if you could pass this on to someone who is and send me a short email telling me their name and telephone number.

Yours sincerely,

Steve Reeves

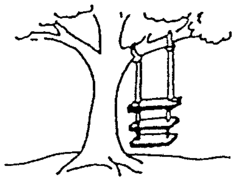
for the ISuRF team:
Lindsay Groves, Ray Nickson
Steve Reeves, Mark Utting

A.2 The enclosure

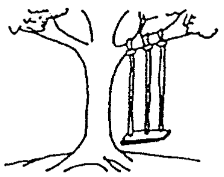
Improving Software using Requirements Formalization

Summary

Formalization allows requirements for custom software to be written precisely and so aids communication. It also allows us to provide tools for checking and reasoning about specifications, so that the goal of correct and reliable software may be more readily achieved. Our project is investigating cost-effective ways of applying formalization in the New Zealand software industry.

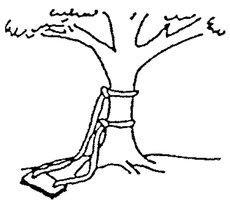

As proposed by the project sponsor.

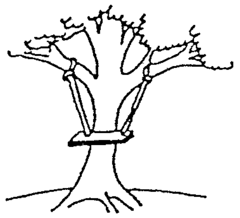
Formal specification is a set of techniques for describing the desired behaviour of a computer system precisely, using standard mathematical concepts such as sets, functions and relations. Typically, a formal specification is written early in the software lifecycle, to capture just the essential requirements of the system and the top-level design of the system. Formal specifications can also be used at the module level, or class level in an object-oriented design. Formal specification techniques can be used by software developers, as well as by companies that develop requirements for customized software then subcontract out the development.


As specified in the project request.

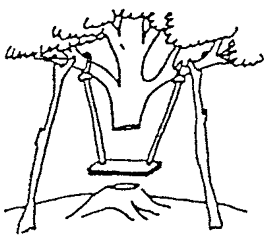
Some of the main advantages of formal specifications are:

- The process of writing a specification typically exposes ambiguities and areas of incompleteness in the informal system requirements specification.
- They can be type-checked and analyzed by various programs to detect errors.
- Errors and areas of incompleteness found during formal specification are found earlier in the lifecycle than with traditional techniques. This reduces the cost of fixing those errors.
- The specification is a precise starting point for detailed design and coding. This can reduce errors during the coding stage and result in higher quality software.
- Several studies have shown that the extra time spent during and coding phases, with a slight overall saving in time.[i]
- The specification makes design review meetings more effective and precise.[ii]
- The specification can guide the development of black-box test suites.
- Specifications at the module or class level can often be translated into run-time assertions within the implementation of the class. This provides a high degree of error detection during the debugging and testing phases of a project.

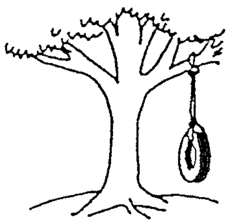

As designed by the senior analyst.


As produced by the programmers.

It seems likely to us that formal specifications can make a significant improvement in the quality and cost-effectiveness of software development. A key attraction is that the specifications are precise enough to be extensively analyzed and manipulated by computer tools, and this can be done early in the software lifecycle. Like static type checking, which was rarely used 30 years ago, but is now widely used and incorporated in standard programming languages, formal specification techniques may be considered to be an essential phase of software development in the future. However, careful investigation of the benefits and limitations of the techniques is needed first.


As installed at the user's site.

Our project is reporting on requirements techniques currently used by New Zealand software developers and on opportunities for introducing new techniques to improve software quality. In addition, a detailed investigation, based on projects from two leading New Zealand software development companies, will determine exactly how formal specifications can be integrated with current industry practice, what tool support is desirable, and how cost-effective the integration is likely to be.


What the user wanted.

The project will pave the way for further technology transfer inadequacies in the currently available support tools for specifications, by developing prototypes of new tools that are better integrated with current industry practice, and by raising awareness of the potential benefits of formalization.

[i] See "CICS/ESA 3.1 experiences" by Mark Phillips in "Z User Workshop, Oxford 1989", Springer-Verlag 1989, and "Using Formal Methods to Develop an ATC Information System" by Anthony Hall in IEEE Software, Vol. 13, No. 2, March 1996.
[ii] See "Formal Methods Light" by Cliff Jones (page 20) in IEEE Computer, Vol. 29, No. 4, April 1996.

B Telephone Questionnaire

Questionnaire for ISuRF project

Person being spoken to (name, address, email):

Basics:

What is the primary business of your company:

Services:	Products:
one-offs	one-offs
customised	batch jobs
mass provision	mass produced

How many people work for the company in New Zealand?

How many, if any, work in software development?

Can you give a few examples of what $\left[\dfrac{\text{products}}{\text{services}}\right]$ your company $\left[\dfrac{\text{produces}}{\text{provides}}\right]$?

Developer or Client?

Does your company develop software as (at least part of) its business?

Yes:

do you do development:

- in response to single contracts
- of 'shrink-wrap' software
- of customised software
  (and so have a close relationship
  with the customer during development
  and maintenance)

No:

do you $\left[\frac{\text{buy}}{\text{lease}}\right]$ customised software $\left[\frac{\text{written}}{\text{tailored}}\right]$ for you?

What kinds of software do you $\left[\frac{\text{develop}}{\text{use}}\right]$? (e.g. embedded, client/server etc.)

What size of systems do you $\left[\frac{\text{develop}}{\text{use}}\right]$?

Describe the phases a typical development (with you as the $\left[\frac{\text{developer}}{\text{client}}\right]$) project goes through.

What proportion of effort is spent, typically, on different phases of a project?

How do you express your requirements?

For developers only:

What methodologies do you use?

What tools do you support your development with?
(e.g. CASE, database tools, interface generators)

What languages and development environments do you use?

How do you record and respond to changing user requirements?

Back to both:

Are there industry/ISO/New Zealand Standards that you have to meet?
(also do you meet any CMM, ISO900.... standards?)

What quality control mechanism(s) do you $\left[\frac{\text{use during your software development processes}}{\text{apply to your contractors}}\right]$
? (e.g walkthroughs, testing regimes etc.)

Clients only:

What do you do when the software you have bought fails to meet your requirements?

How do you tell when this has happened?

What proof that requirements have not been met would you typically be able to provide when this happens?

Finally:

Would you be available and happy to meet face-to-face to talk things over further if we wanted to?

Would you prefer we made future contact via letter?

Thanks.

Interviewer's name:

C Evening Post Editorial, Tuesday March 9, 1999

Incis syndrome needs fixing, and quickly

Question: how do you make a bundle? Say, millions of dollars more than you said you'd charge your client? Answer: devise a computer system for a New Zealand government department.

The Integrated Crime Computer System – the computer system designed to do work once done by coppers, clerks and record-keepers – is one such example. Due for completion in 1997, three years after it was conceived, the $98 million system will have cost at least $118 million by the time it's fully commissioned. At latest estimates, that's sometime next year. Because of delays, police have been forced to commit $3 million to upgrade the old Wanganui computer to make it Year 2000 compliant, and no sooner than that's done the expensive ageing computer based in the River City will be made ready for the scrapheap and Incis will be commissioned.

It's been suggested that Incis suppliers IBM will wear as much as $50 million of the eventual cost. It's to be hoped that contractual arrangements ensure that will happen. In the meantime, belief in the integrity of government computer tendering mechanism must surely be suspended. It seems New Zealand public servants have been well and truly stitched up by a belief in computer reliability, an acceptance of failure, and a vanity about computer design know-how. How else could governments be so open to salesmen who sell things that don't go?

The police computer fiasco isn't the first. The sorry tale of government agencies befuddled by the computer business includes the Accident Compensation Corporation, which scrapped two systems in the middle of the decade and managed to get a third system going only after spending what's understood to be about $90 million. The National Library spent $8.5 million on a dud system, but was able to recover $5.2 million last year after junking the venture in 1996. And the police have had other woes to deal with: their $10 million Card system, which won a reputation for ignoring emergency calls and dispatching fire trucks to wrong addresses, finished up costing $14.4 million.

Incis defies belief. However, an inquiry being sought by Labour and New Zealand First into the way taxpayers' money is being tipped into a seemingly bottomless Incis pit is likely only to confirm doubts about the paucity of advice governments get about computer designs and reveal how New Zealand's been done over in the tendering processes. It's hard to escape the conclusion that having got their foot in the door, computer companies can exploit an apparent lack of knowledge. Once they're in, they then have the leverage to squeeze money from clients in order to make reality of dreams. Squealing over costs isn't going to do much good, and neither, in the Incis case, is an inquiry.

The effect of the Incis debacle is that despite soothing utterances by Police Minister Clem Simich, confidence about police effectiveness has taken a dive. Incis is sucking up money that should be spent crime-fighting. MPs traditionally rattle crime-busting sabres in election year, but this year that sound has an increasingly hollow ring. The criminals whose success depends on ham-strung police must be laughing into their drinks.

Voters, for whom Incis seems no more that another acronym denoting government ineptness, may take more notice when criminals strike and they're unlucky enough to be involved. If victims take the view that $118 million would buy a lot of bobbies on the street, they'd be dead right.

Reproduced by courtesy of The Evening Post, Wellington.