JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE
J. Softw. Maint. Evol.: Res. Pract. 2003; **15**:41–59 (DOI: 10.1002/smr.265)

Research

Test processes in software product evolution—a qualitative survey on the state of practice

Per Runeson[*,†], Carina Andersson and Martin Höst

Software Engineering Research Group, Department of Communication Systems, Lund University, Box 118, SE-221 00 Lund, Sweden

SUMMARY

In order to understand the state of test process practices in the software industry, we have conducted a qualitative survey, covering software development departments at 11 companies in Sweden of different sizes and application domains. The companies develop products in an evolutionary manner, which means either new versions are released regularly, or new product variants under new names are released. The survey was conducted through workshop and interview sessions, loosely guided by a questionnaire scheme.

The main conclusions of the survey are that the documented development process is emphasized by larger organizations as a key asset, while smaller organizations tend to lean more on experienced people. Further, product evolution is performed primarily as new product variants for embedded systems, and as new versions for packaged software. The development is structured using incremental development or a daily build approach; increments are used among more process-focused organizations, and daily build is more frequently utilized in less process-focused organizations. Test automation is performed using scripts for products with focus on functionality, and recorded data for products with focus on non-functional properties. Test automation is an issue which most organizations want to improve; handling the legacy parts of the product and related documentation presents a common problem in improvement efforts for product evolution. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS: qualitative survey; verification and validation; industry practice; interviews; evolution

1. INTRODUCTION

Verification and validation activities take a substantial share of project budgets. Early rules of thumb devoted 50% of the time schedule to testing [1], and no great breakthroughs seem to have changed this dramatically. An earlier survey, with a focus on lead-time consumption, concludes that there is

[*]Correspondence to: Per Runeson, Software Engineering Research Group, Department of Communication Systems, Lund University, Box 118, SE-221 00 Lund, Sweden.
[†]E-mail: per.runeson@telecom.lth.se

Received 30 June 2002
Revised 25 September 2002

a significant shift of the main lead-time burden from programming to integration and testing, when distributing systems [2]. Verification and validation (V&V) are the activities performed during a software development project to ensure that the right system is developed (validation) and that the developed system is right (verification) [3]; hence half of the time is spent checking that what is done during the other half is correct. V&V activities primarily include inspection and testing, and in this survey we focus on the testing part.

The high ratio of time and effort spent on verification and validation seems to be particularly true for product evolution, where development efforts from earlier releases of the product can be reused, but all functionality has to be verified and validated in every release. In order to understand the state of practice of the processes in industry, a qualitative survey was launched, covering software development departments at 11 companies in Sweden of different sizes and application domains, all evolving products continuously, either releasing new versions of existing products, or new product variants. The survey is conducted through workshop and interview sessions, loosely guided by a questionnaire scheme. The departments range from six to 200 developers, in domains of communications, image processing and support systems. In this paper, these departments are referred to as 'organizations'. They are selected based on availability, but we ensured that there is sufficient variation with respect to size, age and application domain, to draw relevant conclusions based on the findings.

The methodology followed in the study is presented in Section 2 and the surveyed organizations are briefly presented in Section 3. The observations and the analysis are reported in Section 4 and, finally, the conclusions are summarized in Section 5.

2. RESEARCH QUESTIONS AND METHODOLOGY

The purpose of the survey is to investigate the current status of verification and validation processes in software companies developing systems in an evolutionary manner. However, as a qualitative survey is not an objective study, but a view of the world seen from the researchers' viewpoints, the approach to the survey as well as the specific research questions are biased by the researchers. In order to enable critical reviews of the observations and conclusions of the study, the viewpoints of the researchers are reported here, leading to the research questions investigated.

2.1. Researcher viewpoints

Below, a few statements summarize the values of the researchers which performed this investigation.

- *Process focus.* The documented process is an important means for communication and capturing experiences. The communication may regard tasks to perform and the progress of the development project. Experience capturing may comprise historical time and defect data, which require a process model as a reference. The term process here is broadly defined to include a variety of methods and other developmental support [4], while the documented process refers to company development manuals and similar documents.
- *Balance between process and people.* All knowledge and experience cannot be captured in a documented process. Software engineering is hence heavily dependent on individuals.

The process is assumed to be more important for large organizations, while people are more important for smaller organizations.

- *Inspections*. It is assumed, and to some extent empirically shown [5,6], that inspections provide efficient means for early defect removal. Inspections are also assumed to contribute to information spreading within a project or an organization.
- *Structured V&V*. It is assumed that a structured approach to V&V would help many organizations and improve their efficiency and effectiveness [7].
- *Product evolution*. Products evolve through their life cycles, which most often are longer than originally planned. Products evolve, either as distinct releases of the same product, or as components in new products in a product family fashion. It is assumed that this is all too often an *ad hoc* process, both with respect to product and process [8].

These viewpoints are further defined in terms of the set of interview questions raised, and might unconsciously impact on the observations and the interpretation of the observations. Hence this open presentation of the view provides the readers with means to arrive at their own interpretation.

2.2. Research questions and method

Based on the researcher viewpoints, and the questions pinpointed for this special issue, we have addressed six research questions.

RQ1. How much is the documented process emphasized by the organizations?
RQ2. Are there any relations between the process emphasis and the characteristics of the organizations or their products?
RQ3. Which criteria govern the selection of the process?
RQ4. Which kinds of evolutionary development exist among the organizations?
RQ5. How is the test automation tailored to support evolutionary development?
RQ6. Which criteria guide the improvement of the process?

In order to address these questions, the survey is guided by qualitative research methodology, hence using a flexible design [9]. This is not intended to be in contrast to the need of quantitative research in software engineering [10,11]. Instead, the methodologies are expected to complement each other. Quantitative methodology is better suited for studies on, for example, specific methods or notations, while qualitative methodology is better suited for broader studies that seek to present overviews or more generalized information. The analysis of RQ1–3 and RQ6 is also reported by Andersson and Runeson [12], with more detailed observations seen from a more general V&V process viewpoint.

A general model for qualitative studies is shown in Figure 1. The data in this study was (a) collected using unstructured interviews, to some extent in the form of group interviews [9]. Data are recorded by two researchers taking notes from the interviews[‡]. Hence some data reduction (b) was conducted during the observations. Further data reduction occurred in a later step based on the topics selected for deeper analysis. The data collection procedures are presented in more detail in Section 2.3.

[‡]The interviews with organizations Phone and Security were recorded by only one researcher.

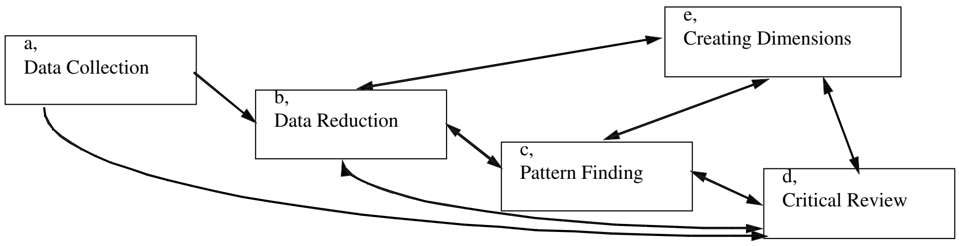

Figure 1. A general model for qualitative studies [13].

The observations were compiled into reports which were sent to the interviewees for feedback, who thus acted as critical reviewers. A third researcher was also a critical reviewer in the analysis (d).

The analysis (c, e) was conducted by using a conceptually clustered matrix [9]. Data was recorded in a matrix in columns related to the issues in the interviews. The data analysis procedures are presented in more detail in Section 2.4.

2.3. Data collection procedures

The survey was conducted in two cycles. First, five organizations were surveyed using the workshop format. The organizations attended a workshop series, organized within a local software process improvement network (SPIN)[§]. The attendants assigned themselves to the workshop based on their interests. The attendants belonged to quality departments or test departments, and most attendants were present at most workshop meetings.

The workshop cycle was conducted as follows.

1. The workshop host presented the company V&V activities. The hosts were free to present any topic, as long as they included a list of strong and weak issues regarding their V&V process.
2. The researchers checked that a list of questions was covered [12] and raised questions that were not voluntarily addressed.
3. The researchers summarized the meeting in a report, which was proof-read by the company representative.
4. The findings were analyzed, compiled into a joint report and fed back to the organizations.

In the second cycle, another six organizations were surveyed using a more direct interview format. Those organizations were approached by the researchers and selected to achieve diversity with respect to size, age of organization, and application domain. This cycle was conducted in a similar manner, except that the only participants at the meeting were the interviewees and the researchers. Finally, within the second cycle, one of the first five organizations was interviewed again (Read, see Table I),

[§]http://www.spin-syd.org.

since they were at a point of major process change at the time of the first interview. The second interview was conducted six months later, when some of the changes had been implemented.

2.4. Analysis procedures

Data analysis was conducted in three cycles. First, the workshop data collected was summarized in a rather informal report, for internal use amongst the workshop participants. Second, an analysis of the survey in general was conducted, as reported in [12]. Thirdly, the analysis was conducted towards the specific focus of test processes for evolutionary software development. The analysis was mainly conducted by two researchers, and a third researcher acted as a critical reviewer.

The procedures can be summarized as follows, with references to Figure 1.

1. The data was structured in a matrix with rows for each participating organization and columns representing different aspects covered during the interview (b). The first set of columns were selected from the areas in the questionnaire checklist (e). The data was primarily taken from the compiled report and secondly from the original notes from each interview.
2. The data in each column was then analyzed; categories were sought and ranked with respect to different criteria (e).
3. The analysis was made based on the matrix; relations as well as lack of relations were sought (c).
4. The analysis was cross-checked by a third researcher, acting as a critical reviewer of the procedures and the analysis (d).

2.5. Validity

In empirical studies, the concept of validity is central [9,11]. The validity analysis seeks to identify threats to the research's validity, and proposes actions to be taken to improve validity. Different models for validity classification exist. Here we adhere to the model originally presented by Lincoln and Guba [9,14].

The model divides threats to validity into three broad headings: *reactivity*, *respondent bias* and *researcher bias*. Reactivity and respondent bias include the risk that the respondent acts differently than normally, for example, acting differently due to the researchers' presence or answering to fulfill the researchers' expectations, instead of answering truthfully. *Researcher bias* refers to the preconditions and assumptions the researchers bring into the situation, which may affect, for example, data collection or analysis.

To reduce threats to validity, different strategies can be implemented, addressing different kinds of threats. A list of such strategies [9] for dealing with different threats is presented below.

- *Prolonged involvement* means that the researcher follows the respondents for a longer period of time, to get acquainted with the studied environment. This strategy is not explicitly used in the current study, although there is a long cooperation between one of the researchers and the studied organizations in general within the local network, reducing the threats of reactivity and respondent bias while, on the other hand, increasing researcher bias.
- *Triangulation* means having multiple sources for the data. In the current study, *observer triangulation* is used, i.e. two researchers are present at almost all interviews. Furthermore, the

interviews include triangular questions: direct questions where the interviewees are asked how they perform their V&V, and indirect questions where they are asked what is good and bad in their V&V process.
- *Peer debriefing and support* refers to having peers cross-check the analysis and act as a coach to the researchers. This strategy was used in this study, having two researchers in the data collection and analysis, and a third reviewer acting as a quality assurance person.
- *Member checking* means returning material to the respondents for feedback. This is used in the current study by reporting the results back to the subjects of the study in writing and in seminars.
- *Negative case analysis* refers to 'playing the devil's advocate'. This strategy is applied in the study, requiring the third researcher try to find alternative cases as an explanation to the variation in the observed data.
- *Audit trail* means keeping a full record of the activities during the study. This is a weakness of the current study, as the interview material is not tape-recorded. However, the researcher view is presented (see Section 2.1), the investigation procedures are openly reported (see Sections 2.3 and 2.4), and data collection is documented, although not publicly available for confidentiality reasons.

The observations presented reflect the survey participants' answers, i.e. it is the organizations' own picture which is presented, with a risk that it is polished. The participants might have given answers, which may not accurately reflect the situations at the associated organizations. However, this risk is limited in the current study, as the participants had nothing to gain from polishing the truth. Furthermore, within the software process improvement network, where most of the surveyed organizations take part, there is a tradition of openness between peers as well as towards the researchers and other external sources. Hence, we cannot find any reason that the survey participants would polish the truth, nor do we believe there is a difference in this respect between the workshop and the interview sessions.

3. SURVEYED ORGANIZATIONS

Our sample in the survey is comprised of departments at 11 Swedish companies. They represent a diverse selection of application domains, product types and company characteristics, although they are not systematically sampled from any larger distribution. As they sometimes are just very small parts of large companies, we refer to them as organizations. The surveyed organizations are listed in Table I in decreasing size order, and referred to by pseudonyms for confidentiality reasons. The terms used to characterize the organizations are defined below.

3.1. Product characteristics

The products developed by the surveyed organizations are characterized along three dimensions.

The *application domains* of the surveyed organizations are of five kinds.

- Radar image processing means a radar system for surveillance of large areas.
- Communication involves networked products as well as wireless communication products.

Table I. Participating organizations.

	Product characteristics			Process characteristics				Organizational characteristics			
Pseudonym	Application domain	Product type	Product value	Process emphasis	Process structure	Evolution strategy	Automation approach	Customer type	Business model	Size	Age
Radar	Radar image processing	Embedded	Non-functional	++	Increment	Variants	Recorded data	Defense	Contract	200	>20
Phone	Communication	Embedded	Functional	+	Increment	Variants	Scripting	Private	Market	150	~15
Network	Communication	Embedded	Functional	+	Increments[a]	Variants	Scripting	Business	Market	120	18
Automation	CASE tools and control	Packaged	Functional	++	Daily build	Versions	Scripting[b]	Business	Market	80	~25
CASE	CASE tools	Packaged	Functional	0	Increment	Versions	Scripting	Business	Market	50	11
Read	Image processing	Embedded	Non-functional	–	Daily build	Variants	None	Private	Market and Contract	30	6
Security	Image processing	Embedded	Non-functional	–	Daily build	Variants	Recorded data	Business	Market	20	5
Monitor	Image processing	Embedded	Non-functional	–	Daily build	Variants	Recorded data	Business	Market	20	3
Product	Support systems	Packaged	Functional	– –	Daily build	Versions	Recorded data	Internal	Contract	7	4
Sales	Support systems	Packaged	Functional	– –	Daily build	Versions	Scripting	Internal	Contract	7	10
SubSales	Support systems	Packaged	Functional	– –	Daily build	Versions	None	Subcontractor	Contract	6	11

[a]One option for projects.
[b]Trial.

- CASE tools refer to software tools for developing other systems, either for general software systems, or for specific instances of systems including hardware components.
- Image processing means products which take pictures and analyze them for different purposes.
- Support systems are business information systems, intended to support different kinds of business.

The *product type* defines how the software is offered to the market. Embedded software refers to software which is embedded in some sort of equipment, including the hardware which the software runs on. Packaged software is sold as a separate unit and is installed by the user on, for example, a PC or workstation platform.

The *product value* may be functional or non-functional. A PC application has primarily, for example, its value in the functions provided to the user, not that the reliability of the software is very high. In contrast in an embedded system, for example for image processing, non-functional properties like the quality of the image or the speed of the image transfer are of primary interest to the user, while the list of functions in such a product is shorter.

3.2. Process characteristics

The processes used in the surveyed organizations are characterized with respect to four dimensions, which are further analyzed in Sections 4.1–4.4:

The organizations *emphasize* the value of the documented process differently. The organizations are classified in five classes ($--$, $-$, 0, $+$, $++$) from no emphasis on a documented process to very much emphasis on an extensively documented process.

The *process structure* is characterized as either daily build or incremental. In *daily build*, all changes are made available to the whole project on a daily basis; in *incremental development*, new versions comprising a set of implemented functions are delivered to integration and system testing at a specified time interval, typically monthly.

The *evolution strategy* is the strategy applied to develop new products, either:

- new *versions* of a product are delivered at various intervals, generally under a new version number (e.g. 1.3, 2.0).
- new *variants* of products are delivered, generally under a new name (e.g. X200, X300).

The test *automation approach* used is either based on recorded data or on scripts. Using recorded data means that *input data* to the product is not taken from the real environment, but from an earlier execution of the system in its environment. Using scripts means running programs that execute the system under test, feeding *events* or *input actions* into the system.

3.3. Organizational characteristics

The surveyed organizations are characterized according to four different aspects.

Customers are either defense, business, private or internal. This variation implies that different kinds of time constraints, market requirements and business models are represented.

- Defense customers comprise large, long-term contracts between customer and supplier.
- Business customers buy products on contract or off-the-shelf for business use.

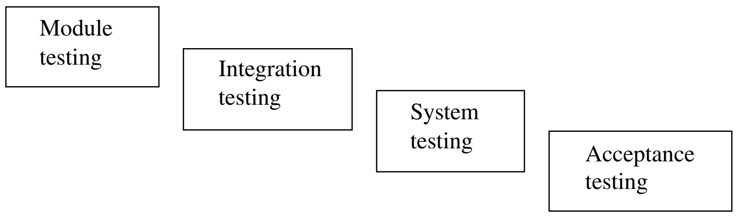

Figure 2. The general test process.

- Private customers buy products for private use off-the-shelf.
- Internal customers use products developed by another department of the company.

Business models are either market, i.e. the products are offered to a wide variety of customers, or contract, i.e. the product is developed and sold to a specific customer.

The *sizes* of the organizations include all engineers involved in the development, including test staff. The companies as such may be much larger, but here only the staff related to the product development departments are counted.

The *age* of the organizations reflect the number of years the surveyed activities in each company have been active.

4. OBSERVATIONS

Observations and subsequent analyses regarding the test processes are reported in this section. The observations are presented structured according to the research questions in Section 2.2.

4.1. Process emphasis

During the workshop and interview sessions, discussions concerning the test process proceeded from a general test process, according to Figure 2, since all of the organizations had some model to follow, though not always documented. All organizations were able to map their activities into this general model, though with varying degrees of formality. The survey displayed a spectrum of process definitions, ranging from a very well-defined process at Radar to a very informal and unemphasized definition at SubSales. Brief characterizations of each organization's process are presented in [12].

One clear observation from the survey is that a documented process is considered an important asset among the large organizations, for testing and for development in general, while smaller organizations rely more on experienced individuals. Table II shows the relationships between organization size and the degree of emphasis of the process.

This might be an obvious observation. However, some thresholds are also observed with respect to size and process emphasis. Organizations smaller than 10 developers have almost no process structure at all. They rely on skilled and experienced people. Organizations of size 20–30 begin to identify the

Table II. Process emphasis versus number of developers.

Process emphasis		Number of developers			
		1–10	20–30	50–100	150–200
Emphasized	++			Automation	Radar
	+				Phone
					Network
	0			CASE	
			Security		
			Monitor		
			Read		
Not		Product			
Emphasized		Sales			
		SubSales			

need for more structure. They start defining templates and some basic procedures. Organizations of size 50–100 rely on rather well-defined templates and common process step definitions. Organizations larger that 150 stress the importance and value of well-defined work instructions.

The two organizations that emphasize the process the most are both somewhat special. Automation is a part of a larger body and has processes in common across the corporation. Radar has maintained high requirements for well documented processes for a long time through its operations in the defense domain.

A common opinion among the organizations is that it is important to increase the visibility of the V&V process and allow the members of the organizations to understand the importance of V&V. Communicating the process is of high priority and often a part of the ongoing improvement work.

In summary, there is a wide range of different attitudes towards the value of the process (RQ1), and the most visible relation to the process emphasis is the size of the organization (RQ2). There seems to be a breakpoint somewhere between 30 and 50 developers in the organization, over which the process is needed to guide and support the work, while below the threshold inter-project communication is sufficient using less formal means.

4.2. Process selection

Some organizations have chosen an explicitly incremental development process model [15]. Subsets of functionality are defined as increments to allow better control over the projects and to reduce risks. Radar, Phone and CASE have such models (see Table III). In Network, the projects may choose an incremental model as one of the options. Other organizations have smaller, less-defined increments, more towards the daily build principle [16,17]. This kind of approach is used by Automation, Read, Security, Monitor, and SubSales. Read and Phone have made pilot use of the extreme programming concept [18] in single projects. Organizations with more emphasized processes tend to use the

Table III. Process emphasis versus process structure.

	Daily build	Increment
Emphasized [0, +, ++]	Automation	Radar Phone CASE Network[a]
Not emphasized [−, −−]	*Security* Read Monitor Product Sales SubSales	

[a]One option for projects.

Table IV. Overview of evolutionary approaches.

Approach	Description	Timescale
Daily build	New versions of the systems are delivered internally	Day(s)
Incremental development	New versions are delivered to integration	Month(s)
Product evolution	New versions of the system are delivered to the market	0.5–1 Year

incremental approach, while the less emphasized processes are of a more daily build character (see Table III). This relation is quite natural as incremental development requires a more well-structured approach, while daily build is the bottom-up solution to avoid big-bang integration problems. The exception, Automation, has a tradition of using daily build for approximately 20 years; this policy was established when the company was still young and small.

The incremental and daily build approaches both introduce evolutionary product development into the internal development process. The principles are the same, although the timescale is different (see Table IV).

Common to the approaches are that new versions are delivered to a stakeholder on a regular basis; in daily build, all changes are made available to the whole project on a daily basis; in incremental development, new versions comprising a set of implemented functions are delivered to integration and system testing at a specified time interval, typically monthly. Product evolution implies new releases are delivered to the market, typically once or twice a year. Using daily build or incremental development provides the opportunity to start testing early on the earlier increments with limited functionality, and thus reduce the cycle time for a release since it allows testing in parallel with development.

Network has tried the incremental approach and found that it was not profitable. The hand-over from development to testing became fuzzy, and software which was not fully unit tested was delivered to system test. Hence, they had to identify multiple failures which were otherwise removed by the developers. The other organization using incremental development did not experience such problems.

In summary, incremental development is used by organizations with an emphasized process, while the daily build approach is primarily used by organizations with less emphasized process (RQ3).

4.3. Product evolution

All of the surveyed organizations develop products that evolve over time. We have identified two distinct approaches to product evolution. Either they release new *versions* of a product, or they deliver new *variants* of products (see Table VI below). Versions are marketed under the same product name, but with a new extension (1.3, 2.0), while variants are marketed as new products under a new name (X100, X300). The new versions and variants must differ substantially from the earlier ones, in particular for organizations with private or business customers as their market. The motivation for buying a new product with embedded software, or upgrading to a new release of packaged software must lie in added value for the user, in terms of better non-functional properties or new functionality. Upgrades are offered to the market, however, not bought by every customer and thus multiple versions of the product have to be supported.

In addition to the major releases, new versions of the software with minor changes (polishing), can be developed. This is referred to as 'product care' by one of the interviewees. In the case of packaged software, 'patches' or 'service packs' can be distributed to customers, while in the case of embedded software, new versions of the software are introduced in the production process gradually. Products already delivered are only updated when very critical faults are detected.

In both the version and the variant cases, a large portion of the software remains the same, and test effort already invested could be saved by either considering software components well tested 'by use' or by automating regression tests. The former approach involves a risk, clearly illustrated by the Ariane 5 failure [19]. The latter approach is used by the surveyed organizations to different extents, but in no case to a very large extent.

From the survey we can conclude that embedded products are mainly developed using the variant strategy, while packaged software is mainly developed using the version strategy, see Table V. The first case is quite natural, as long as dynamically downloading new software in embedded products is not permitted to the user. The second case is not bound by these kinds of technical constraints. However, there are indications that changes are under way. In the Phone case, a new version of the software is offered to the users of a specific model of an embedded product, and new variants are planned to allow dynamic download of, for example, Java programs. This trend is indicated by an arrow in Table V. Further, the Sales/SubSales product is built on a general software framework, which is intended to be used as a basis for other variants of products, i.e. approaching the variant approach for a packaged software product. This trend is indicated by the second arrow in Table V. More details on the different evolution strategies are presented in Table VI.

In summary, there are two kinds of evolutionary strategies, variants and versions (RQ4), where the former is primarily used for embedded products and the latter for packaged software, although the borderline is under change. In addition, effort is spent on 'product care'.

Table V. Product type versus evolution strategy.

	Variant	Version
Embedded	Radar Read Security Monitor Phone Network	 →
Packaged	 ←	Automation Product Sales SubSales CASE

Table VI. Brief overview of the evolution strategy for each company.

Radar	Radar develops three variants of their product for three different customers. From a software perspective, they are not managed as a product line [8]. Instead, the development chain was branched for each variant and is not merged again. However, information about identified faults in one variant are fed back to the development of the other variants.
Phone	Phone uses the variant approach to develop new products. The same hardware and software architecture is used in several variants of the product.
Network	Network develops products using a planned product line approach [8]. They develop products whose appearance and usage are quite different, while internally they are built on the same hardware and software platform.
Automation CASE	Automation and CASE both release new versions of their product regularly. As the customers have to pay for the new release, or foresee problems with the introduction of new releases, not all customers upgrade for every release, implying that many subsequent releases have to be supported in parallel.
Read Security Monitor	Read, Security and Monitor have the same origin. The products of the three organizations share processor and optical devices for image processing, while the analysis algorithms and the functionality of the products are very different. The common parts are rather stable and not further developed very much, and are primarily considered a platform.
Product Sales SubSales	Product and Sales (and hence SubSales) deliver new releases of a software package at pre-determined dates. The customer is internal or semi-internal, thus avoiding the requirements of real market customers. In the Product case, the delivery date is negotiable if the development runs out of time. In the Sales case, features are negotiable but not delivery dates, as the system supports the sales department, and one frequent reason for a new release is that a new price list is introduced from a certain date.

Table VII. Product type versus product value.

	Embedded	Packaged
Non-functional	Radar Read Security Monitor	
Functional	Phone Network	Automation Product Sales SubSales CASE

4.4. Test automation in product evolution

Test automation is mentioned by most of the surveyed organizations as an improvement area. Most organizations automate testing to some extent, but they are not satisfied with the level of automation.

The automation approach is different, depending on the product characteristics presented in Section 3.1. We concluded from the data that there is a tendency towards believing that non-functional properties constitute the key value of the embedded products, and that functionality constitutes the key value of the packaged software (see Table VII). The two exceptions, *Phone* and *Network*, where functionality constitutes the value for embedded products, are mature communication products. In these domains, the characteristics are taken for granted by the users, and the functionality constitutes the competitive advantage for new products. Hence the products are on the borderline of having their focus on functionality from the user point of view, even though the developers still consider the non-functional properties to be of great value.

Two different approaches to test automation exist among the surveyed organizations. Either the execution of features is automated by scripts running the application, or the inputs to the systems exist as recorded data from a real environment. Using scripts means running programs that execute the system under test, feeding events or input actions to the system. For example, the communication products are tested by running a script which sends commands to the program, which in the operational environment comes from the user. Using recorded data means that input *data* to the product is not taken from the real environment, but from an earlier execution of the system in its environment. For example, the image processing products are tested using previously recorded images.

Product value is connected to the type of automation approach chosen. Table VIII shows an overview of approaches chosen by the organizations. Products with functional focus are tested using scripts and products focused on non-functional properties are tested using recorded data.

Organizations that do not use an automation approach have a good reason for not doing so in most cases.

Table VIII. Product value versus automation approach.

	Recorded data	Scripting	None
Non-functional	Radar Security Monitor		Read
Functional	Product	Phone Network Automation[a] CASE Sales	SubSales

[a]Trial.

- The Read product involves a feed-back loop, changing the parameters for data recording based on the analysis of data, e.g. to compensate for different light conditions. Hence, earlier recorded data are not realistic enough to use as test data.
- SubSales is primarily responsible for module tests, while Sales is responsible for the integration and system tests. There is a planned improvement effort to also run the automated tests at the module level.

Although most organizations already perform some form of automated tests, this issue is among the most mentioned of the improvement areas. A key issue regarding automation, mentioned by *Sales* is product stability. Investments in test scripts are large and are only worthwhile for a long-lasting product or product line.

In summary, there are two kinds of test automation among the surveyed organizations, recorded data for products with the focus on non-functional properties and scripting for products with the focus on functionality (RQ5).

4.5. Test process improvement

Most of the surveyed organizations report an intention to improve the test process, although the approach is not very structured, nor emphasized very much. Examples of improvement initiatives are presented in Table IX.

The improvement approaches taken are very different. The only rationale for choosing a certain approach that we could observe during the survey, is that the selection depends on the persons involved.

One specific issue observed in product evolution regarding process improvement is what happens when, for example, the requirements specification is improved for the functionality of a new release, while the requirements for the old parts remain badly specified, as it is too costly to update the requirements for the legacy part of the product. Testing based on the requirements can then be improved for the new parts, while the basis for the regression testing of the legacy part remains bad.

Table IX. Brief overview of improvement initiatives for each company.

Radar	Radar works on improving the efficiency of system testing. Test planning methods based on facto rial designs [20,21] are used and found to be efficient. Improvements are driven by a joint university–industry research project.
Phone	Phone has recently turned towards incremental development to reduce risks and allow flexibility in which functionality is delivered in certain products. Testing can be started much earlier and thus identify integration problems sooner.
Network	The Network quality department makes process guidelines available to the projects but does not require a specific approach to be chosen. This liberal attitude may be related to the fact that the company is not part of a larger corporation like Radar and Phone.
Automation	Automation has, based on the outcome of the workshop behind this survey, initiated a project to store test cases in a database, together with some structuring and test time information, to allow more efficient regression testing. The initiative is taken by interested individuals.
CASE	CASE has, after its recent merger, produced a new and more structured set of test documents. The new set is based on the best practices of the two merged organizations.
Read	Read has assessed their test process during the period of this survey and designed a new one. The person responsible for process improvement was working on documenting and structuring the process in a previous employment; hence he chose this approach in his new affiliation. However, as the company was restructured during the period, and reduced in size, only some basic parts of the proposed process are introduced.
Security	Security has used the test process improvement (TPI) approach [22] to assess their current status and find improvement areas.
Monitor Product Sales SubSales	Monitor, Product and Sales/SubSales are working on hands-on improvement actions, like document standards etc. The companies are the smallest and have no tradition of working with defined processes or company standards.

In summary, the approach chosen for process improvement is not based on any of the observed characteristics, but seems to be ruled by the involved persons' background and experience (RQ6).

5. SUMMARY AND CONCLUSIONS

Verification and validation of software systems take a substantial share of project resources as well as lead-time for a project. In product evolution, this is assumed to be particularly true, as the old parts of the products have to be regression tested to verify that they still work as intended with new functionality added. In order to understand how the verification and validation processes are constituted at organizations of different size and different application domains, a qualitative survey was launched. As all of the organizations perform product evolution, this is an aspect of particular focus in the survey.

The first observation from the study is that there is a wide range of attitudes towards the value of the process, and that larger organizations tend to emphasize the process more than smaller organizations.

There seems to be a breakpoint somewhere between 30 and 50 developers in the organization, with respect to the emphasis of the documented process (RQ1 and RQ2).

The evolution approach taken among the organizations is either incremental (internal release cycles of month(s)) or daily build (internal release cycles of day(s)). We can conclude that organizations which emphasize the documented process use incremental development, while organizations which have a less emphasized process use daily build (RQ3). One organization differs from the pattern. This organization has been using daily build for 20 years, and has turned towards more process emphasis later during its growth and mergers with other companies.

We can conclude that there are two different kinds of product evolution, for embedded software and for packaged software (RQ4). New releases of embedded software products constitute a new product in a product family. The products are sold to new customers or to existing ones; in both cases, a new ‘thing’ is delivered to the customer. In the packaged software case, new versions are mostly delivered as upgrades to existing customers or as new deliveries to new customers, although there are indications of other combinations of the two. As all customers are not willing to upgrade immediately, different versions have to be supported in parallel. In addition to new variants and new versions, product care is performed which implies minor changes to an embedded product or patches to packaged software.

Test automation is considered an area where there are potential savings in product evolution, and they are exploited to some extent (RQ5). We have identified two kinds of test automation, recorded data automation and scripting automation. We conclude that products which focus on non-functional properties are primarily tested using recorded data, while products with a focus on functionality are tested using scripts for automation. The two observed exceptions from this rule are both mature products with a focus on non-functional properties, in which the users take the non-functional properties for granted. Hence, the products are on the borderline and maintain their focus on the functionality from the user point of view, even though the developers still consider the non-functional properties be of greater value.

We can conclude from the survey that there is no observed relation between the product characteristics, nor the process focus on which approach is taken for improvement. It seems to be very much dependent on the persons involved, their experiences and their personal viewpoints (RQ6).

In summary, there is a set of specific issues regarding product evolution observed among the surveyed organizations, although the potential savings regarding, for example, regression testing are not fully exploited. This paper, and the workshops and interviews behind it, contribute to an increased awareness of the variation factors in product evolution, a first step towards an improved test process for product evolution.

ACKNOWLEDGEMENTS

The researchers are grateful to the participating companies for their contribution. Thanks also to Daniel Karlström and Thomas Olsson at the Department of Communication Systems at Lund University, for reviewing an earlier version of the analysis report and being involved in discussions on the topic. Thanks to the reviewers and editors of the special issue for their valuable comments. This study was partially funded by The Swedish Agency for Innovation Systems (VINNOVA) under a grant for The Center for Applied Software Research at Lund University (LUCAS).

REFERENCES

1. Brooks FP. *The Mythical Man-Month*. Addison-Wesley: Reading MA, 1975.
2. Bratthall L, Runeson P, Adelswärd-Bruck K, Eriksson W. A survey of lead-time challenges in the development and evolution of distributed real-time systems. *Information and Software Technology* 2000; **42**(13):947–958.
3. Boehm B. Software engineering: R&D trends and defence needs. *Research Direction in Software Technology*, Wegner P (ed.). MIT Press: Cambridge MA, 1979.
4. Lindvall M, Rus I. Process diversity in software development. *IEEE Software* 2000; **17**(4):14–18.
5. Basili VR, Selby RW. Comparing the effectiveness of software testing strategies. *IEEE Transactions on Software Engineering* 1987; **13**(12):1278–1298.
6. Porter AA, Siy HP, Toman CA, Votta LG. An experiment to assess the cost-benefits of code inspections in large-scale software development. *IEEE Transactions on Software Engineering* 1997; **23**(6):329–346.
7. Kit E. *Software Testing in the Real World: Improving the Process*. Addison-Wesley: Reading MA, 1995.
8. Bosch J. *Design and Use of Software Architectures: Adapting and Evolving a Product-line Approach*. Addison-Wesley: Reading MA, 2000.
9. Robson C. *Real World Research* (2nd edn). Blackwell: Oxford, 2002.
10. Juristo N, Moreno AM. *Basics of Software Engineering Experimentation*. Kluwer: Norwell MA, 2001.
11. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers: Norwell MA, 2000.
12. Andersson C, Runeson P. Verification and validation in industry—a qualitative survey on the state of practice. *Proceedings International Symposium on Empirical Software Engineering*. IEEE Computer Society Press: Los Alamitos CA, 2002.
13. Lantz A. *Intervjuteknik* (Interview methods—in Swedish). Studentlitteratur: Lund, 1983.
14. Lincoln YS, Guba EG. *Naturalisitic Enquiry*. Newbury Park and London, 1985.
15. Sommerville I. *Software Engineering*. Addison-Wesley: Reading MA, 2001.
16. Cusumano MA, Selby RW. *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People*. Free Press, 1995.
17. McConnell S. *Rapid Development: Taming Wild Software Schedules*. Microsoft Press, 1996.
18. Beck K. Embracing change with extreme programming. *IEEE Computer* 1999; **32**(10):70–77.
19. European Space Agency. *Ariane 5 Flight 105 Inquiry Board Report No 33–1996*. http:// ravel.esrin.esa.it/docs/esa-x-1819eng.pdf [26 September 1996].
20. Cohen DM, Dalal SR, Parelius J, Patton GC. The combinatorial design approach to automatic test generation. *IEEE Software* 1996; **13**(5):83–88.
21. Berling T, Runeson P. Application of factorial design to validation of system performance. *Proceedings 7th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*. IEEE Computer Society Press: Los Alamitos CA, 2000; 318–326.
22. Koomen T, Pol M. *Test Process Improvement, A Practical Step-by-step Guide to Structured Testing*. Addison-Wesley: Reading MA, 1999.

AUTHORS' BIOGRAPHIES

Dr Per Runeson is an Associate Professor in Software Engineering at the Department of Communication Systems, Lund University, Sweden, and has been the leader of the Software Engineering Research Group since 2001. He received a PhD from Lund University in 1998, and a MSc in Computer Science and Engineering from the same university in 1991. He has five years of industrial experience as a consulting expert in software engineering. Dr Runeson's research interests concern methods and processes for software development, in particular methods for verification and validation, with special focus on efficient and effective methods to facilitate software quality. The research has a strong empirical focus. He has published more than 40 papers in international journals and conferences and is the coauthor of a book on experimentation in software engineering.

Carina Andersson received an MSc in Engineering Physics with Industrial Economy from Lund University, Sweden, in 2001. She is currently a PhD student in software engineering at the Department of Communication Systems, Lund University. Her main research interests include methods for effective and efficient verification and validation processes for software development.

Dr Martin Höst received an MSc from Lund University, Sweden in 1992 and a PhD in Software Engineering from the same university in 1999. He is currently an Assistant Professor in Software Engineering at the Department of Communication Systems, Lund University. He has published papers in international journals and conferences and is the coauthor of a book on experimentation in software engineering. Dr Höst's main research interests include software process improvement, empirical software engineering, using quantitative and qualitative research, early evaluation of software process change proposals, and computer simulation of software development processes. In addition to research, Dr Höst is a programme leader for the MSc program in Information and Communication Systems at Lund University.