C. Wohlin, A. Gustavsson, M. Höst and C. Mattsson, "A Framework for Technology Introduction in Software Organizations", Proceedings Software Process Improvement Conference, pp. 167-176, Brighton, UK, 1996.

A Framework for Technology Introduction in Software Organizations[1]

Claes Wohlin[*, 2], Anders Gustavsson[**], Martin Höst[*] and Christer Mattsson[**]

[*] Dept. of Communication Systems
Lund Institute of Technology
Lund University
Box 118
S-221 00 Lund
Sweden
E-mail: (claesw, martinh)@tts.lth.se

[**] Q-Labs AB
IDEON Research Park
S-223 70 Lund
Sweden
E-mail: (agu, cma)@q-labs.se

Abstract

Software process improvement is difficult. The benefits of each new process change proposal or new technology is hard to judge. A systematic approach to introduction of new technology and change of software processes is presented in this paper. The objective is to provide a framework for evaluation of the impact of a software process change. The framework provides a basis for evaluating changes in a software organization, and it is based on the identification of three major types of evaluation: surveys, experiments and case studies. These types are further divided into seven separate methods of evaluating process change proposals and hence managing change. Examples of usage of the three types and the seven methods with respect to commercial projects are provided. It is concluded that cost-effective change of software processes is difficult, and that the proposed methods provide a framework for organizations trying to cope with process change.

Keywords

Process improvement, change management, technology transfer, technology introduction, case study, software experiment, impact analysis.

1. This work is a part of the PERFECT project and it is sponsored by the Swedish National Board for Industrial Technical Development (NUTEK). PERFECT stands for Process Enhancement for Reduction of software deFECTs and it denotes the ESPRIT project 9090 funded by the CEC in which the following organizations participate: CAP Gemini Innovation, Daimler Benz, LGI, Q-Labs, Robert Bosch, Siemens Norway, and University of Kaiserslautern. In PERFECT, the department of Communication Systems at Lund University is a subcontractor of Q-Labs.

2. Parts of this work has been carried as acting professor at the Department of Computer and Information Science, Linköping University, Sweden.

1.0 Introduction

1.1 Background

The introduction of changes in a software organization must be made systematically and based on informed decisions. An organization developing software cannot adopt every trend or new technology emerging. In this context, a technology is defined as processes, methods, techniques and tools in a software organization. Thus, it is assumed to include changes to any process, for example, processes related to people issues, for example change of processes supporting skill development of the personnel, as discussed in P-CMM [Hefley95]. New methods and tools are presented regularly in a rapid pace, but an organization developing software cannot be expected to change in this pace. It is necessary for an organization to be able to judge, which changes are suitable and hence beneficial for them, economically, technically or in other terms.

To ensure cost-effective and controlled process change, a framework for systematic process change is needed. This type of framework can be viewed as one important part of a general software process improvement programme. The objective of this paper is to present a framework for systematic evaluation of software technology. Furthermore, to provide software organizations with a means for evaluating process change and hence introduce of software technology in a systematic way. The specific methods within the framework are not new, but the comprehensive framework is. The framework relates the methods to each other in a systematic way.

The proposed framework can be used for both technology introduction into an organization developing software and to evaluate new software technology emerging from research centres, for example universities or research departments within large software organizations.

Systematic process improvement is first discussed to put the proposed framework into context. Thereafter, the application of the framework is discussed. Finally, some conclusions are presented.

1.2 Systematic process improvement

Traditionally, process improvement in industry has been a means of changing the used software processes in order to achieve some stated goals with respect to, for example, quality, time to market and productivity. Unfortunately, most of the changes have been done based on beliefs or assumptions that a specific change will affect the product or process in a positive direction relative to the goals. The results have often led to the opposite with unpredictable costs for the technology change and unpredictable quality, time to market and productivity. The solution to this is to focus on systematic process improvement.

Systematic process improvement refers to goal-oriented measurement, [Basili94a], and a controlled way of introducing process change, with predictable outcome in terms of the above parameters. Thus, goals and a strategy for evaluating change must be determined Based on the strategy and the goals, specific measures can be derived to enable evaluation of the actual impact of the proposed change. The formulation of goals must

be company specific, and the strategy must be tailored to the organizational needs and the people involved in the technology change, [Gustavsson95].

Most approaches to software process improvement are primarily focused on process assessment and identification of potential areas for improvement. This is the main approach taken in, for example, CMM [Paulk95], Bootstrap [Kuvaja94] and SPICE [Rout95]. A recent attempt to systematise and evaluate the support for people working with software development, and evaluate areas for improvement, is proposed through the People-CMM [Hefley95].

The single most important factor in software development is the skill of people [Boehm81]. Thus, new technology should only be introduced as a means for helping people develop their personal skills. Technology should be for the people, which implies that changes must be introduced in a way that people feel involved and in a positive atmosphere. A key issue to succeed with process improvement is thus to have commitment from management, but perhaps even more important to ensure that people would like to change, i.e. people wants to start using the new technology.

The framework proposed in this paper may seem technology-oriented, but the objective is really to provide means for demonstrating usefulness of new technologies both to management and to the people involved. In our opinion, a key issue, to get commitment from everybody involved, is to start in a small scale and to show the potential benefits, and to illustrate for the people that using the new technology actually is rewarding and stimulating. Thus, the framework should act as a facilitator for process change by providing a systematic approach to change, which allows us to demonstrate, market and get commitment before introducing a change in a whole organization.

The assessment is an important part, but support is also needed for introducing changes and supporting people and organizations in their objective to improve. Some different approaches exist in this area too, for example, the Personal Software Process [Humphrey95], and the Quality Improvement Paradigm and Experience Factory [Basili94b]. The PSP focuses on how individuals can work continuously with improving their own way of working, and the experience factory approach primarily is aimed at the organizational level.

All of these approaches are important and have their advocates, but it is not, in our opinion, possible to state that one of them is better than the other. The approaches can, for example, be combined to enjoy the benefits from several of them. It is, for example, possible to adopt the Quality Improvement Paradigm and use Bootstrap for characterization of the organization.

The approaches discussed so far do not have any explicit support for evaluation of new technology proposals or for technology transfer in an organization. Some possible evaluation methods for this are: surveys, experiments and case studies. These are techniques that help us understand, evaluate, guide, control, predict and improve our software processes. The need for experimentation in software engineering is stressed in [Basili86], although it recently has been argued that if a change is not obviously good then we should not bother [Davis96]. The latter view seems a little naive based on the complexity of large scale software development, and in particular since, for example, a process change may improve software quality, but it may have a negative impact on time to market. How do we judge and evaluate this issue without investigations?

The objective of this paper is to put the evaluation methods into context, and provide a framework to structure the relationship between the methods.

2.0 Technology introduction

2.1 Introduction to the framework

A systematic approach to technology introduction is needed and illustrated in Figure 1. The focus here is on evaluation of technology, while CMM or other models can be applied for assessing an organization. The “Evaluate Technology” box in Figure 1 is expanded in Figure 2. It is important to stress that iteration and feedback within and between the different parts are essential although not indicated in the outline.

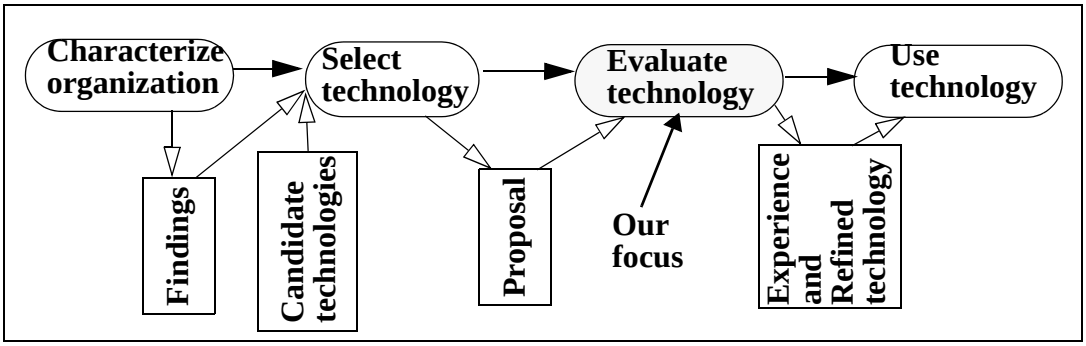

FIGURE 1. A scenario of technology introduction for process improvement.

A major problem in the improvement process within software engineering is the inability to reuse experiences from other organizations, and in particular specific values concerning increase in quality or productivity and also reduction of time to market. Success stories presented in the literature cannot just be transferred into another organization. A technology which is beneficial in one environment may be less rewarding in another context, hence most organizations must strive to create their own experience base, [Basili94b]. To enable the creation of an experience base without taking chances, it is necessary to identify a number of possible methods for evaluation of software technology. A framework with a number of possible methods is depicted in Figure 2. The methods indicate an increase in the following: cost, confidence in the outcome and similarity in terms of context in comparison to the ordinary software development environment. This is further discussed in Section 2.3. It should also be noted that the methods are possible methods and they should not be interpreted as a number of steps that have to be conducted after each other in the presented order.

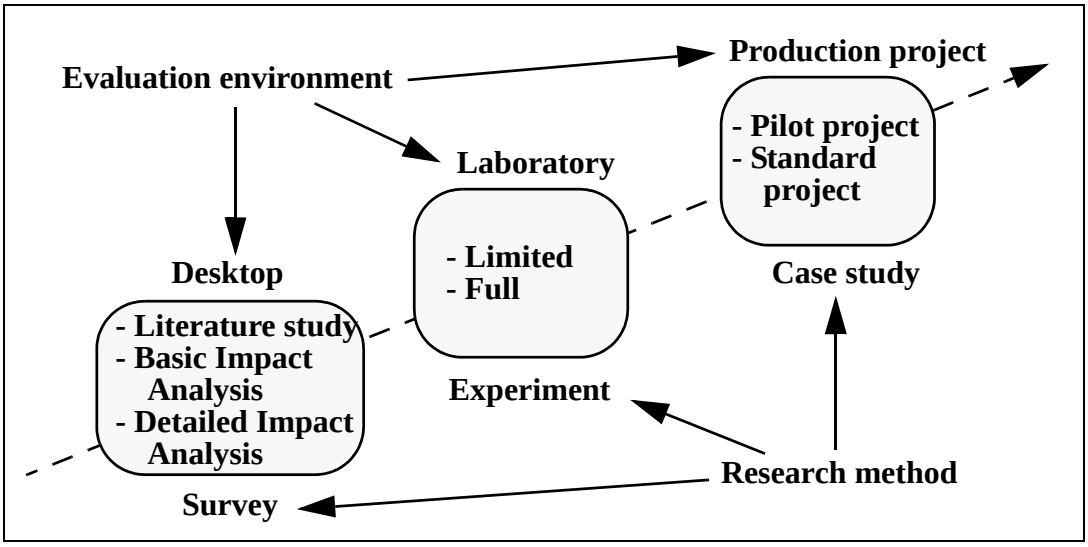

FIGURE 2. A number of possible methods to evaluate software process change.

The evaluation environment may be of three different basic types, and the environment is closely related to the research method applied, see Figure 2. These are all further elaborated below.

The approach discussed here with a scientific and systematic approach to process improvement is supported by others. For example, in [Basili86] experimental software engineering is discussed, the need for a scientific approach is stressed in [Fenton94], and experiments as a means in software engineering is thoroughly described in [Pfleeger94]. A systematic approach to case studies is presented in [Yin84], and it is further elaborated for software engineering in [Kitchenham95]. The state of the art in experimental software engineering is summarized in [Votta95].

2.1.1 Desktop/Survey

A survey can be based on available literature, experiences stored in the experience base or it can be based on subjective expert judgement. The survey results can be used for a desktop evaluation, where the information gathered is combined using common sense or more formally models of the execution.

The models used for prediction can either be very simple (basis impact analysis), implying that the values collected are used almost as is, or the models could form a more advanced prediction system (detailed impact analysis). This can take into account dependency between different attributes, for example, time to market and one or several product quality attributes. A method and some simple models for impact analysis prior to change are presented in [Höst95]. Thus, the survey methods are:

- Literature study

  A literature study is the expected starting point when evaluating a specific process change proposal. It provides an opportunity to review the current state of the art and often also the best available practice in the area. The information collected can either by judged subjectively or, if possible, objectively based on published figures

and expected impact in our specific environment. The lessons learned from the literature study must be combined with the actual environment in which the technology change is considered.

- Basic impact analysis

  Information is collected, primarily from earlier experiences and expert judgements, for an overview of the problem, i.e. we do not examine how, for example, a specific method affects all different parts of a process. Instead, the focus is on overall change in performance.

- Detailed impact analysis

  In this particular case, the desktop study is performed in-depth. For example, experts from different areas are interviewed for their opinions. The overall impact of, for example, a new method is derived, using a set of models of project execution and the detailed information. No individual is in control of the overall result of the evaluation.

Example:

One example of a change proposal is to introduce a formalized inspection process into an organization's software development processes [Fagan76]. This example is presented in three different parts, one describing desktop evaluation, another laboratory evaluation, and a third the implementation in production projects.

The first step is to search the literature for articles and books about software inspections. This particular area has been relatively well covered by the literature, both in terms of books, for example [Gilb94], and through articles both describing inspection techniques, for example [Fagan76], and experience and experiments when using inspections, for example [Porter94] and [Wohlin95].

After the literature study, we assume that an organization decides to perform a basic impact analysis to better understand the characteristics and impact of introducing formal software inspections. This means that a number of people who are able to overview the complete development process could be consulted. Their estimates can then be used to form a first insight on the efficiency of the proposed technique.

The detailed impact analysis would be based on the estimated impact on the sub-processes. Therefore, a more extensive data collection must be performed. Here, a number of experts from each of the different parts of the process, such as high level design, design and test could be consulted. These experts can give their estimates of the new inspection technique's impact on their respective parts of the process. When this data has been collected mathematical models can be used to predict the overall impact [Höst95]. The output from the models forms the basis for a decision whether to continue the evaluation, and which method to apply next, or if the technique should be dismissed.

2.1.2 Laboratory/Experiment

A laboratory means that the techniques proposed are evaluated in an off-line environment. The objective is to resemble the actual software development environment, but the resulting software is not expected to be delivered to any customer. A laboratory

means experimenting and trying to come to a conclusion regarding the effectiveness of a particular software technology. Experiments, and in particular the design and analysis of them, are discussed in [Pfleeger94]. Experiments can be divided into two major types.

- Limited experiment

  A limited experiment is restricted in size or in scope, in particular probably including evaluation of an isolated technology. The experiment can, for example, be an investigation of an inspection method which is taken out of context from the development process.

- Full experiment

  A full experiment is when the investigation is made as a normal project, although within a laboratory environment, i.e. the requirements regarding delivery dates and so forth are not realistic.

Example:

An inspection technique can be evaluated in laboratory experiments in a number of different ways. One way to construct a limited experiment is, for example, to let a number of independent groups of people inspect the same material with a known number of faults in it. In this limited experiment the efficiency of the inspection technique can be assessed. This limited experiment would involve people using the new technique in a stand alone experiment as if the technique was part of a complete project.

To estimate the inspection technique's impact in the complete development process, and hence to put it into the context of the whole process, a full experiment can be conducted where the technique is applied in an off-line project. This would involve people using the new technique as if the project was a production project, but with the limitation that it is not carried out with the constraints of executing it on-line, for example changing requirements. This means that the organization can repeat the full experiment to be able to tune the technology to its needs.

Examples of limited experiments with inspection can be found in [Porter94] and [Wohlin95]. Examples of experiments can be found in [Basili94c] and [NASA94]. In [NASA94], it is for example reported from a full experiment where a new object-oriented technique together with Ada was used for development in parallel with the development of the same product using standard techniques and FORTRAN.

2.1.3 Production project/Case study

A case study is normally a study conducted in parallel with the execution of a project. It should be planned in advance, but we have less control over the execution than in an experiment. We are normally external observers of a "real" software project. Case studies in software engineering are discussed in [Kitchenham95].

The introduction of a new software technology into software production projects can be made in two major different ways.

- Pilot project

  A pilot project is a project which is conducted within an organization with the objective both to deliver a final product and to evaluate, for example, some specific methods before disseminating them to the whole organization. The latter requires, of course, that the pilot project is a success and that the proposed method is judged as being valuable.

- Standard project

  The study is conducted on a real project. The project is run as a normal project within the development organization. The primary objective is to deliver the software product. The standard project implies that the new technology has been incorporated in the standard process, and hence is an ordinary part of the software development process.

Finally, when the organization has evidence or confidence based on desktop evaluations and laboratory experiences the new technology, for example a formal inspection technique, can be introduced in a pilot project. This gives a more final confirmation on the characteristics and efficiency of the technology in a project with production constraints, time pressure, etc. Special attention should be given to evaluating how the inspection technique affects the project, and in particular the product quality. Based on the experiences of the pilot project, an eventual standard implementation can be planned.

In [NASA94] and [Tann93] it is reported from different pilot projects. In [NASA94], it is, for example, reported that Independent Verification &Validation (I V&V) techniques have been evaluated in two different pilot projects. These pilot projects did not result in any standard implementation.

An implementation of the inspection technique in a standard project means that the inspection technique has been introduced, as an ordinary part, into the standard process used in the project.

2.2 Summary of the methods

The main characteristics of the different evaluation methods are summarized in Table 1.

TABLE 1. Characteristics of evaluation techniques

Type of evaluation	**Characteristics**
Literature Study	No software process is executed. Available literature regarding the technology is gone through.
Basic Impact Analysis	No software process is executed. Interview based (with real staff and external experts) - limitations in population and in depth.
Detailed Impact Analysis	No software process is executed. Interview based (with real staff) in depth. Models for reliability, productivity etc. are added.

TABLE 1. Characteristics of evaluation techniques

Type of evaluation	Characteristics
Limited Experiment	Off-line. Context independent. Isolated and limited process executed.
Full Experiment	Off-line. Simulated context. Full process executed (no reductions).
Pilot Project	On-line designated projects. Production context. Full process executed, which means that the standard process is used and complemented with the new technology. Product focus. Technology change focus.
Standard Project	On line. Production context. Standard process executed (the new technology is incorporated in the standard process). Product focus. Process compliance and follow-up focus.

It is quite obvious from Table 1 that technology evaluation is not for free. It does take time and effort to evaluate technology, but in large scale software development organizations where the life time of the software products span 20-30 years, we cannot just change due to trends. Change must be determined based on informed decisions. Technology change is crucial to stay on the competitive edge of software development, and hence it is necessary to invest in evaluations.

2.3 Interpretation of the framework

The different methods of evaluating software technology before introducing them into an organization may be used either in order or a certain evaluation may be skipped due to some reason, for example, experience indicating success with a high probability. It is, however, important to realize that the number of factors increases with the different approaches, some of them positive and other negative. Going from literature studies to standard projects, it is obvious that:

- the cost for performing the evaluation increases,
- the similarity in terms of context to software production increases,
- the confidence in the evaluation increases.

These facts are important to take into consideration when discussing evaluation of new software technologies, and hence process change.

Although, Figure 2 does not indicate iteration, it is important to use the methods for process change evaluation with feedback. The experience gained must be fed back to improve the evaluation method. Assuming, for example, that we start with a basic

impact analysis, and then a limited experiment is conducted prior to changing a certain software process on a standard project in the organization, then it is essential to check the outcome of the different evaluation methods towards each other. Otherwise valuable information of improvement is not made use of.

3.0 Application of the framework

3.1 Introduction

The possible methods for technology evaluation can be used for two main purposes:

- technology change in an organization developing software,
- technology evaluation prior to transferring the technology for broader use.

The former includes introduction of technology, which may be accepted in other applications, marketed by a specific company or transferred from a research environment. This implies that the methods are used by an organization developing software. The latter item means using the framework in a research environment, which could be either a university or a research department within a company.

The software development organization adopting the framework can use it to evaluate, for example, new CASE tools, new techniques and methods, either marketed by other companies or new research results being available. The organization can choose to use all methods in the framework, or to use some of them or in extreme cases to fully deploy a new technique directly. The objective of the methods is to provide guidelines for technology change without being prescriptive. All seven methods in the framework are relevant for an organization developing software.

The research centres can probably not use all seven methods in the framework, although some exceptions exist, for example, the Software Engineering Laboratory [NASA94]. Most research centres can, however, often only use the four first methods in the framework. A desktop evaluation can always be done and most research centres can perform limited experiments, but it is in most cases infeasible to expect universities and research departments to perform software production projects. Therefore, it is important to accept and adopt an approach where the research centre preprocess new findings or proposals by performing limited experiments and that the new research results are thoroughly evaluated by the software development organization prior to adoption of the technique. This is of great importance as the research centres perform their desktop evaluation and limited experiment based on their environment.

The objective must, however, be that the limited experiments give some indication of the usefulness of a particular research result. Most results presented in the software engineering literature can probably be viewed as limited experiments, hence making it extremely difficult to draw any general conclusions about the advantages and disadvantages of a particular software technology. The development of large software systems, which have a long life time in operations, is a complicated and complex business, hence it is not likely that we based on a limited experiment are capable of evaluating all the possible implications when introducing a process change into a development proc-

ess to be used in a large software projects. Thus, limited experiment should be viewed as a means to get indications and to motivate further investigations on a larger scale.

It should also be noted that by applying the proposed methods with feedback, it should be possible to create a set of activities that can be replicated, from literature studies to standard projects. This requires documentation of:

- how the new technology has been applied,
- the characteristics of the environment,
- the results of the different methods, which should include the correspondence between the outcome of the different methods.

Adopting the proposed framework for technology evaluation and applying it should give better control, when introducing new technologies. It is not possible to change the way software is developed, without some indications that the change actually is an improvement.

3.2 Using the framework

A scenario for using the framework to investigate and evaluate new technology is summarized in Table 2. The reliability of the information refers, of course, to the reliability of applying the new technology in a standard project.

It must be noted that a suitable first step is always to study the available literature to obtain a baseline concerning the state of the art in the area and also to get some information about best practices. Based on the characteristics of the information obtained from the literature study, and the judgement of the reliability of the result, it is possible to formulate some possible entry criteria into the rest of the framework, see Table 2.

TABLE 2. Scenario for entry criteria into the framework.

Characteristics of information source	**Reliability of information**	**Type of evaluation proposed**
New ideas and research results.	Low - ideas with limited spread and use found in journals & at conferences.	Basic Impact Analysis
Some scattered examples - “success stories”	Low to medium - some reports on the success of introducing these techniques. Results cannot be repeated.	Detailed Impact Analysis
Consistent data from other organizations/research institutes	Medium - some evidence that the results can be achieved in repeated experiments. Some information on how the experiments were carried out is available. The results are often context independent.	Limited Experiment

TABLE 2. Scenario for entry criteria into the framework.

Characteristics of information source	Reliability of information	Type of evaluation proposed
Data and trends from other organizations/research institutes	Medium to high - evidence for a trend. Repeated experiments carried out. Experiments confirmed by other organizations. The results are often context independent.	Full Experiment
Experiences from own organization	High - experiments within the same organization, context, projects etc. are known. Results can be repeated.	Pilot Project
Experiences from similar projects in the same organization and application domain	Very high - technique is used in the same context, by similar projects within the own organization. Hypothesis can be formulated with high probability that it can be fulfilled.	Standard Project

An organization can based on business goals or markets requirements understand that the quality of their product has to be increased. Based on experiences from previous projects, stored in an experience base, the organization has a clear view and understanding of where in the process the quality problems arise. A number of candidate techniques that may solve their problems have been identified both internally in the cooperation and externally in, for example, the research community via publications at conferences and in journals. The question is of course how well the candidate techniques fulfil the quality goals and how confident we can be that the stated goals can be reached by the proposed technologies.

For technologies where only “success” stories of other organizations or research results exist it is recommended to make a basic impact analysis to evaluate the proposed technology within the organization. When more in depth results are presented, a detailed impact analysis can be performed. The desktop evaluations allow the organization to easily and with a limited cost sort out technologies not appropriate for the organization. The next step to take is to perform some experiments out of the context of the development organization. Some organizations provide this kind of experiments allowing for other to gain from these experiments [NASA94]. The final step is to introduce the technology in the production context, testing whether the selected technology meets the stated hypothesis. This could be done in two steps, pilot projects and finally by introducing the technology into standard projects.

As experience is gained, the organization has to evaluate and examine the appropriateness of the technology in its own organization with respect to their goals, and compare with experience from other technologies. In addition to the discussed framework and its methods in this paper, systematic process improvement requires a set of activities in the range from commitment building, training of staff to methodology development and production of guidelines and handbooks, etc.

It is by no means easy to evaluate process change proposals. A number of problems has to be considered carefully, when performing empirical studies in software engineering, for example problems related to the validity of the evaluation:

- Reuse of experiences across different organizations

  Principal results from an experiment are more easily reused between organizations than specific figures. Reuse between organizations only becomes feasible if the experiments are documented and reported so that the can be replicated by others, and even so the figures may differ due to factors either out of control or stochastic variations.

- Student vs professional

  The ability to transfer results from an experiment conducted solely by students to industrial use can be questioned, and the result must be interpreted accordingly.

- Limited vs full

  A limited experiment, for example only conducting an inspection out of the development context, can be questioned as it is difficult to judge how it will work in the full context.

- Toy vs realistic

  A realistic software engineering experiment is bound to become large and costly, hence minor studies are often conducted. It is, however, difficult to determine how a small experiment scale up to a realistic environment.

These are challenges in experimental software engineering, not obstacles and problems.

4.0 Conclusions

It is obvious that introduction or change of software technologies is difficult and expensive. Software organizations cannot introduce new technologies based on anecdotal evidence of successes. Therefore, a systematic approach to process improvement is needed. This includes methods to evaluate the technology proposal. The evaluation is an integral task when taking a decision whether or not to introduce a particular technology.

In this paper, a number of methods for software technology introduction have been examined. Technology has been used as a collective term for processes, methods, techniques and tools. The proposed methods form a framework which can be used when either evaluating new technologies in the industry or for technology transfer from research centres. The proposed framework is not supposed to be prescriptive. On the contrary, it should provide decision-makers with guidelines concerning possible measures to take when evaluating new technologies.

Based on prior knowledge and experience, the framework can be entered at a suitable level and some methods can be skipped due to confidence in the technology being evaluated. It is, however, important to acknowledge that the methods which are used highly influence the cost and the confidence in the evaluation. These aspects must be taken into account when adopting the proposed framework.

5.0 References

[Basili86] Basili, V., Selby, R. and Hutchens, D., “Experimentation in Software Engineering”, IEEE Transactions on Software Engineering, Vol. SE-12, No. 7, pp. 733-743, 1986.

[Basili94a] Basili, V., Caldiera, G. and Rombach, H.D., “The Goal Question Metric Approach”, in Encyclopedia of Software Engineering, Vol. 1, editor: J. J. Marciniak, pp. 528-532, John Wiley and Sons, New York, 1994.

[Basili94b] Basili, V., Caldiera, G. and Rombach H.D. “Experience Factory”, in Encyclopedia of Software Engineering, Vol. 1, edited by J.J. Marciniak, pp. 469-476, John Wiley & Sons, New York, 1994.

[Basili94c] Basili, V. and Green, S., “Software Process Evolution at the SEL”, IEEE Software, Vol. 11, No. 4, pp. 58-66, 1994.

[Boehm81] Boehm, B., “Software Engineering Economics”, Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1981.

[Hefley95] Hefley, W.E., Miller, S., Curtis, B. and Konrad, M., “The People Capability Maturity Model: Status and Progress”, Proceedings 5th International Conference on Software Quality, pp. 471-483, 1995.

[Davis96] Davis, A.M., “Eras of Software Technology Transfer”, IEEE Software, March 1996, pp. 4 and 7, 1996.

[Fagan76] Fagan, M., “Design and Code Inspections to Reduce Errors on Program Development”, IBM Systems Journal, Vol. 15, No. 3, pp. 219-248, 1976.

[Fenton94] Fenton, N., Pfleeger, S. and Glass, R., “Science and Substance: A Challenge to Software Engineers”, IEEE Software, July, pp. 86-95, 1994.

[Gilb94] Gilb, T. and Graham, D., “Software Inspections”, Addison Wesley, Reading, Massachusetts, USA, 1994.

[Gustavsson95] Gustavsson, A. and Mattsson, C., ”The PERFECT Approach to Continuous Improvement”, Proceedings ESI-ISCN '95: Measurement and Training Based Process Improvement, 1995.

[Humphrey95] Humphrey W.S., “A Discipline for Software Engineering”, Addison-Wesley, 1995.

[Höst95] Höst, M. and Wohlin, C., “Impact Analysis of Process Change Proposals”, in Software Quality, pp. 311-323, edited by M. Ross, C.A. Brebbia, G. Staples, J. Stapleton, Computational Mechanics Publications, Southampton, United Kingdom, 1995.

[Kitchenham95] Kitchenham, B., Pickard, L. and Pleeger, S.L., “Case Studies for Method and Tool Evaluation”, IEEE Software, July 1995, pp. 52-62, 1995.

[Kuvaja94] Kuvaja, P. and Bicego, A., “BOOTSTRAP - A European Assessment Methodology”, Software Quality Journal, Vol. 3, No. 3, 1994.

[NASA94] McGarry, F, Page, G.P. and Basili, V., “Software Process Improvement in the NASA Software Engineering Laboratory” Technical Report, CMU/SEI-94-022, Software Engineering Institute, December 1994.

[Paulk95] Paulk, M.C. “The Evolution of the SEI´s Capability Maturity Model for Software”, Software Process: Improvement and Practice, Vol. 1, No. 1, pp. 3-15, 1995.

[Pfleeger94] Pfleeger, S., “Experimental Design and Analysis in Software Engineering Part 1-5”, ACM Sigsoft, Software Engineering Notes, Vol. 19, No. 4, pp. 16-20, Vol. 20, No. 1, pp. 22-26, Vol. 20, No. 2, pp. 14-16, Vol. 20, No. 3, pp. 13-15, Vol. 20, No. 4, pp. 14-17, 1994-1995.

[Porter94] Porter, A.A. and Votta, L.G., “An Experiment to Assess Different Defect Detection Methods for Software Requirements Inspections”, Proceedings IEEE International Conference on Software Engineering, pp. 103-112, 1994.

[Rout95] Rout, T.P. “SPICE: A Framework for Software Process Assessment”, Software Process: Improvement and Practice, Vol. 1, No. 1, pp. 57-66, 1995.

[Tann93] Tann, L-G., “OS32 and Cleanroom”, Proceedings First European Industrial Symposium on Cleanroom Software Engineering, Copenhagen, Denmark, 1993.

[Votta95] Votta, L.G. and Porter, A.A., “Experimental Software Engineering: A Report on the State of the Art”, Proceedings 17th International Conference on Software Engineering, pp. 277-279, 1995.

[Wohlin95] Wohlin, C., Runeson, P. and Brantestam, J., “An Experimental Evaluation of Capture-Recapture in Software Inspections”, International Journal of Software Testing, Verification and Reliability, Vol. 5, No. 4, pp. 213-232, 1995.

[Yin84] Yin, R.K., “Case Study Research Design and Methods”, Sage Publications, Beverly Hills, California, USA, 1984.