SOFTWARE—PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2004; **34**:1311–1344
Published online 23 August 2004 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/spe.615

Packaging software process improvement issues: a method and a case study

Tony Gorschek*,† and Claes Wohlin

School of Engineering, Blekinge Institute of Technology, P.O. Box 520, S-372 25 Ronneby, Sweden

SUMMARY

Software process improvement is a challenge in general and in particular for small- and medium-sized companies. Assessment is one important step in improvement. However, given that a list of improvement issues has been derived, it is often very important to be able to prioritize the improvement proposals and also look at the potential dependencies between them. This paper comes from an industrial need to enable prioritization of improvement proposals and to identify their dependencies. The need was identified in a small- and medium-sized software development company. Based on the need, a method for prioritization and identification of dependencies of improvement proposals was developed. The prioritization part of the method is based on a multi-decision criteria method and the dependencies are identified using a dependency graph. The developed method has been successfully applied in the company, where people with different roles applied the method. The paper presents both the method as such and the successful application of it. It is concluded that the method worked as a means for prioritization and identification of dependencies. Moreover, the method also allowed the employees to discuss and reason about the improvement actions to be taken in a structured and systematic way. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: software process improvement; prioritization; dependency mapping; packaging; decision support; industrial case study

1. INTRODUCTION

The production of high-quality software with a limited amount of resources, and within a certain period, is the goal of most software producing organizations. They vary from large corporations with thousands of engineers, to small ones with only a handful. All of them, regardless of location, size or even success rate, are largely dependent on their software *processes*‡ to reach their goals.

*Correspondence to: Tony Gorschek, School of Engineering, Blekinge Institute of Technology, P.O. Box 520, S-372 25 Ronneby, Sweden.
†E-mail: tony.gorschek@bth.se
‡A sequence of steps performed for the purpose of producing the software in question (IEEE-STD-610).

Received 10 November 2003
Revised 4 February 2004
Accepted 8 March 2004

It stands to reason that continuous evaluation and improvement of an existing process (Software Process Improvement—SPI [1]) is crucial to ensure that the organization is successful in its pursuits of quality, and in order to be effective enough to stay competitive in the world of business.

The work presented in this paper introduces a structured way in which software development practitioners (whose organization is subject to SPI efforts), and SPI practitioners alike, can make dependency adherent prioritizations of identified improvement issues (findings from process assessments). The goal is to give small- and medium-sized enterprises (SMEs) a tool to focus their SPI efforts, and not to present 'yet another method' as such. The work presented here should complement already existing SPI frameworks with a modular addition intended to minimize some of the main issues with SPI efforts identified in the literature (see Section 2.1).

The paper is structured as follows. Section 2 gives some background information pertaining to SPI concerns and the motivation behind the work presented in this paper. Section 3 introduces the 'Dependency Adherent Improvement Issue Prioritization Scheme' or 'DAIIPS' for short, and the techniques on which it stands. Section 4 presents a study using DAIIPS in an industry SPI effort. The results from the industry study are subsequently validated through an additional study performed in academia, and the results from the two studies are compared for validation purposes. Section 5 presents the discussion and the conclusions.

2. SPI—RELATED WORK AND MOTIVATION

An SPI scheme is usually based in four fairly straightforward steps: 'evaluation of the current situation', 'plan for improvement', 'implement the improvements', 'evaluate the effect of the improvements', and then the work takes another cycle (see Figure 1, inspired by [2]).

Most well-known process improvement (quality improvement/assurance) frameworks are based on this general principle. From the Shewart–Deming 'Plan–Do–Check–Act' paradigm [3] and Basili's

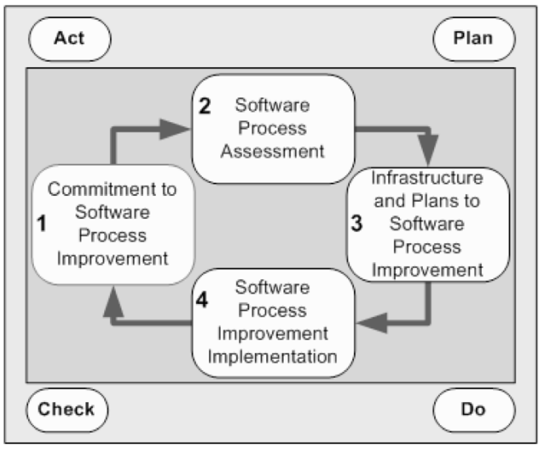

Act
Plan
2 Software Process Assessment
Commitment to Software Process Improvement 1
3 Infrastructure and Plans to Software Process Improvement
4 Software Process Improvement Implementation
Check
Do

Figure 1. Generic process improvement scheme.

'Quality Improvement Paradigm' [4], to standards like CMM [5], CMMI [6], ISO/IEC 15504 (also known as SPICE) [7] and the Software Engineering Institutes IDEAL SPI Guide [8].

Looking at the frameworks mentioned above (primarily CMMI, CMM, SPICE and IDEAL) they have been used for SPI over a number of years (except for CMMI) and there are quite a lot experiences reported from industry.

The main issues seem to be *cost* and *time*. An assessment–improvement cycle is often rather expensive and time consuming [9]. A typical SPI cycle using, e.g., CMM can take anything from 18 to 24 months to complete [10] and demands much resources and long time commitments in order to be successful. In addition to being time consuming, many view extensive SPI frameworks as too large and bulky to get an overview of and to implement [2,11,12].

However, this is not the same as saying that frameworks like CMMI are inapplicable in general. Organizations with time and resources available report high return on their investment over an extended time period (results indicate both lower costs and higher customer satisfaction) [2,11]. The main issue here is whether or not a small- and medium-sized enterprise has the ability to commit to a large-scale SPI project spanning a long time period as far as the work and pay-off are concerned.

There are examples of attempts to adapt larger SPI frameworks to be more suitable for SMEs. The IMPACT project [13] reports an initiative to use elements from proven technologies like CMMI and SPICE to implement a lightweight SPI framework. Another example is presented in [14] where IDEAL is adjusted for use in SMEs. The work presented in this paper is meant to add to this effort of making SPI available for SMEs.

2.1. SPI concerns and DAIIPS goals

In addition to the factors mentioned above, which may have a negative impact on SPI efforts (pertaining mainly to SMEs), i.e. (I) time (long-term work, long-term gain) and (II) cost/resources (the nature of large SPI frameworks imply commitment of much resources over an extended period of time), there are a number of more general critical aspects (not directly linked to SMEs) presented in the literature [15–19]. Some of the main factors are[§] (III) commitment (to the SPI effort by management, middle management and the staff, e.g. engineers), (IV) focus (on the SPI effort with clear and well-defined goals) and (V) involvement (in the SPI work by staff).

If we look at DAIIPS the first and foremost motivation for development of the framework was to give organizations with limited resources for SPI a chance to choose *what to do first* based on their needs.

The need for this was first recognized in relation to cooperative SPI work conducted at DanaherMotion Särö AB (see Section 4), a medium-sized company about to undertake process assessment and process improvement work of their requirements engineering process. With limited resources available it was crucial that the improvements be manageable in size and time, thus not posing a threat against their core activities (the business of producing income-generating products).

The idea was to prioritize and package improvement issues in smaller and more manageable groups based on how important they were to the organization, and the dependencies between the improvement issues. This division of a potentially large amount of improvement issues (depending

[§]In addition to these there are other factors that influence the success of SPI activities. However, these are not elaborated upon here as they are beyond the scope of this paper.

on what was found during the software process assessment phase) was to help organizations take small steps towards quality improvement at a manageable rate and with a realistic scope. This addresses the factors of (I) time, smaller packages of improvement issues can be implemented and evaluated faster (the results are felt in the organization sooner rather than later), and (II) cost/resources in terms of smaller steps each cost less and large resources do not have to be committed over long periods of time. By dividing a potentially large amount of improvement issues into smaller packages for implementation one could argue that it is easier to define clear and manageable goals for the SPI activities, e.g. results occur sooner and there are a limited number of issues that are addressed (speaking of aspect IV). Many SPI efforts fail before they start, i.e. an assessment is made, then the work stops and no real improvements are made. A lack of follow-through is usual, and this may be due to several of the reasons mentioned above, of which commitment of time and resources is not the least [9]. This is the motivation behind DAIIPS, to offer SMEs a framework to choose what to do, i.e. limit the commitment to a manageable level.

DAIIPS is directly dependent on the involvement, contribution and expertise of the personnel involved in the SPI activity, namely a selection of representatives from the organization (as are most SPI paradigms). This is the nature of DAIIPS, i.e. all the decisions (regarding priority and dependencies) are made by the people working with the current process in the organization. The direct nature of the involvement in the work and decision-making enables the SPI work to be secured in the minds of the very people that the SPI activities influence the most, i.e. relating to aspects III and IV.

DAIIPS is not an absolute method, which should be followed to the letter, rather a framework for gathering data, formatting the data and presenting the results in a way that should ultimately act as decision support for SPI activities.

2.2. DAIIPS and modularity

As mentioned earlier, DAIIPS is not meant to compete with any SPI model, in fact DAIIPS is just a structured way to prioritize and check dependencies amongst the improvement issues already identified using a software process assessment (SPA) tool, e.g. CMM, CMMI, SPICE or IDEAL. Thus, the prerequisite for using DAIIPS is that an assessment has been made and that the improvement issues are documented and explained in such a way that the constituents of the SPI targeted organization can understand them. Given this restriction almost any assessment framework/standard/method can be used, including lightweight methods [20].

3. DAIIPS—AN OVERVIEW

DAIIPS consists of three basic steps, (A) prioritization, (B) dependency mapping, and (C) packaging (illustrated in Figure 2). Steps A and B are performed in sequence at a workshop, while step C is performed at a later stage.

As mentioned earlier, improvement issues identified during the SPA stage of the SPI effort are used as input. Each of the steps is described in further detail below.

3.1. Prioritization (step A)

The idea behind prioritization of the improvement issues in DAIIPS is to introduce an explicit choice amongst the issues identified during assessment, i.e. all improvement issues are channeled into the

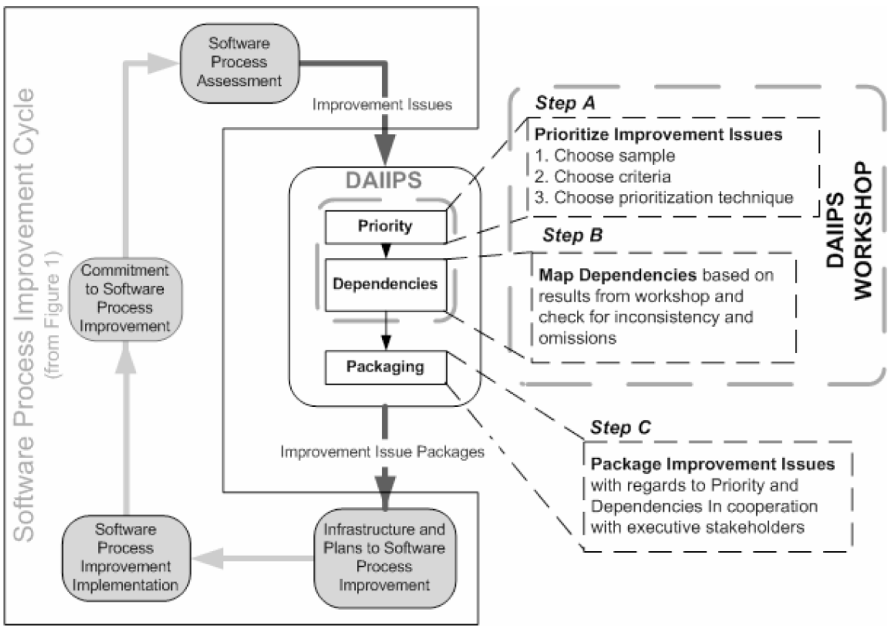

Software Process Improvement Cycle
(from Figure 1)
Software Process Assessment
Improvement Issues
DAIIPS
Priority
Dependencies
Packaging
Improvement Issue Packages
Commitment to Software Process Improvement
Software Process Improvement Implementation
Infrastructure and Plans to Software Process Improvement
Step A
Prioritize Improvement Issues
1. Choose sample
2. Choose criteria
3. Choose prioritization technique
Step B
Map Dependencies based on results from workshop and check for inconsistency and omissions
Step C
Package Improvement Issues with regards to Priority and Dependencies In cooperation with executive stakeholders
DAIIPS WORKSHOP

Figure 2. DAIIPS overview.

prioritization unordered and with no 'importance' attributes attached after the assessment. A number of substeps should be performed in prioritization (see Figure 2), each described below.

For an example of the prioritization (step A) and results from an industry case, see Section 4.3.2.

3.1.1. *Sample*

There should be a conscious decision behind the choice of which stakeholders should be invited to participate in the prioritization step. The sample could be based on a sampling technique—which technique depends on applicability in the case at hand. One alternative could be to invite all of the participants of the SPA performed earlier (if this is a manageable number of people). The participants of the SPA should have been selected in line with some plan (although this cannot be assumed). However, if the SPA sample is deemed good enough it can be reused, saving time both in regards to selecting the sample and introducing the sampled subjects to the SPI activity. There are quite a lot of sampling techniques available, see [21,22] for examples.

The sample size is also important and once again it should be possible to look at the sample size used during the SPA. The sample should be large enough to make it possible to generalize, i.e. a large variability in the population (typical for a software development organization) demands a larger sample.

Quota sampling and stratified random sampling [21] are both based on having elements (strata/groups) from which a representation is wanted, e.g. programmers, middle management, testers and so on. Through sampling, certain views (agendas) can be premiered over others, i.e. by having an overrepresentation of a certain group (role) in the sample. For example, by having an overrepresentation of developers, weight is added to their influence on the prioritization.

Regardless of how the sampling is conducted, it should be a conscious action with regard to the consequences it may have on the prioritization.

For an example of sampling performed in an industry case, see Section 4.2.1.1.

3.1.2. *Criteria*

There are characteristics of the process that we want either to minimize or to maximize. Quality, time, cost and risk are three examples of criteria. When prioritizing improvement issues it is done in relation to a criterion. If quality is the criterion (improve/maximize quality) the prioritization is skewed in one direction, if time is the criterion other aspects may be more important. Which criterion is chosen must be up to the SPA team (including management), and should be based on strategic goals of the organization subject to the SPI as well as the findings of the SPA.

The choice of criteria should be conveyed to the participants beforehand (and consensus reached about what a criterion means) in order for the prioritization to be done with the same focus in mind.

For an example of criteria formulation in an industry case, see Section 4.2.1.1.

3.1.3. *Prioritization techniques*

Several techniques can be used for the prioritization of improvement issues. The main idea is to have a structured way in which prioritization can be performed in the same manner (technique) and with the same intent (criterion) by all participants. Furthermore the results from the prioritization should be comparable to each other, enabling a compilation of the results.

In this paper we briefly present one prioritization method used in our study (see Section 4), and mention three more. Note that the Analytical Hierarchy Process (AHP) is just one of several available prioritization techniques that may be used.

The AHP [23] is a method using scaled pair-wise comparisons between variables, as illustrated in Figure 3. Here the variables are i and j and the scale between them denotes relative importance. The importance ratings can be seen in Table I.

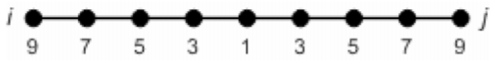


Figure 3. AHP comparison scale.

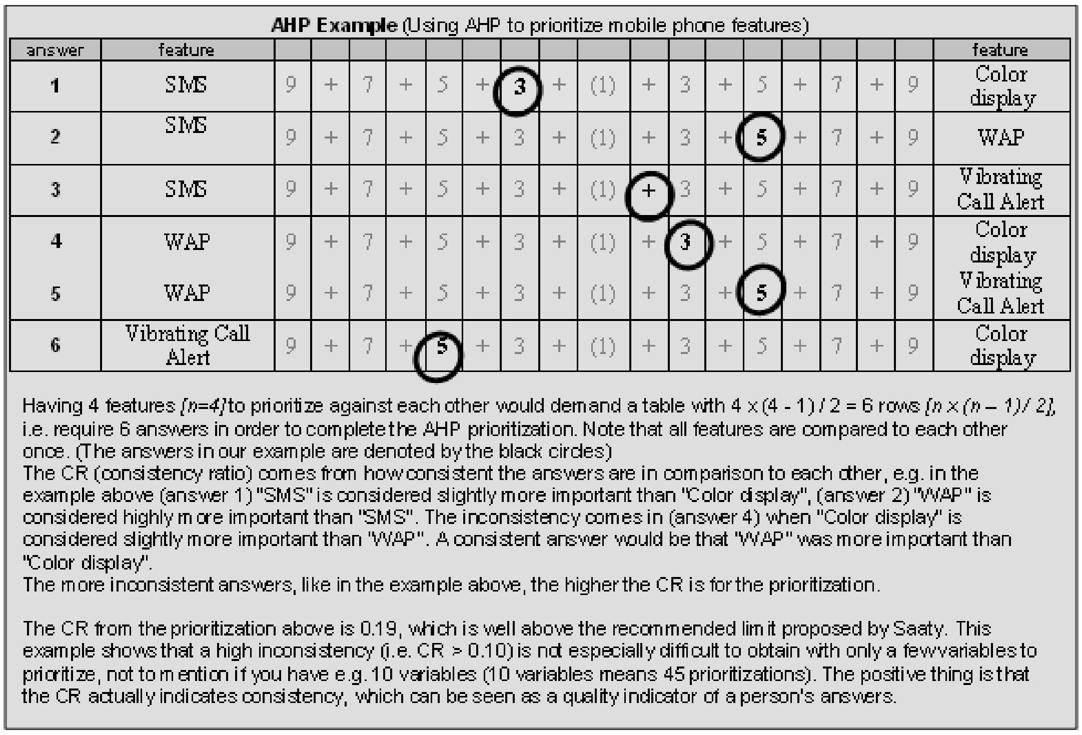
As the variables have been compared the comparisons are transferred into an $n \times n$ matrix with their reciprocal values (n is the number of variables). Subsequently the eigenvector of the matrix is computed. The method used for this is called *averaging over normalized column* and the product is the *priority vector*, which is the main output of using AHP for pair-wise comparisons.

AHP uses more comparisons than necessary, i.e. $n \times (n - 1)/2$ comparisons, and this is used for calculating the consistency of the comparisons. By looking at the *consistency ratio (CR)* an indication

Table I. AHP comparison scale.

Relative intensity	Definition	Explanation
1	Of equal importance	The two variables (i and j) are of equal importance
3	Slightly more important	One variable is slightly more important than the other
5	Highly more important	One variable is highly more important than the other
7	Very highly more important	One variable is very highly more important than the other
9	Extremely more important	One variable is extremely more important than the other
2, 4, 6, 8	Intermediate values	Used when compromising between the other numbers
Reciprocal	If variable i has one of the above numbers assigned to it when compared with variable j, then j has the value 1/number assigned to it when compared with i. More formally if $n_{ij} = x$ then $n_{ji} = 1/x$	

of the amount of inconsistent and contradictory comparisons can be obtained. In general a CR of ≤ 0.10 is considered to be acceptable according to Saaty [23], but a CR of >0.10 is often obtained. There has been some debate as to the applicability of results that have a CR of >0.10, see [24,25], and this is an ongoing debate. A rule of thumb is that a CR of ≤ 0.10 is optimal, although higher results are often obtained in the real world. Further details about AHP can be found in [23,26].



AHP Example (Using AHP to prioritize mobile phone features)

answer	feature																		feature
1	SMS	9	+	7	+	5	+	[✓ 3]	+	(1)	+	3	+	5	+	7	+	9	Color display
2	SMS	9	+	7	+	5	+	3	+	(1)	+	3	+	[✓ 5]	+	7	+	9	WAP
3	SMS	9	+	7	+	5	+	3	+	(1)	[✓ +]	3	+	5	+	7	+	9	Vibrating Call Alert
4	WAP	9	+	7	+	5	+	3	+	(1)	+	[✓ 3]	+	5	+	7	+	9	Color display
5	WAP	9	+	7	+	5	+	3	+	(1)	+	3	+	[✓ 5]	+	7	+	9	Vibrating Call Alert
6	Vibrating Call Alert	9	+	7	+	[✓ 5]	+	3	+	(1)	+	3	+	5	+	7	+	9	Color display

Having 4 features *[n=4]* to prioritize against each other would demand a table with 4 x (4 - 1) / 2 = 6 rows *[n x (n – 1)/ 2]*, i.e. require 6 answers in order to complete the AHP prioritization. Note that all features are compared to each other once. (The answers in our example are denoted by the black circles)
The CR (consistency ratio) comes from how consistent the answers are in comparison to each other, e.g. in the example above (answer 1) "SMS" is considered slightly more important than "Color display", (answer 2) "WAP" is considered highly more important than "SMS". The inconsistency comes in (answer 4) when "Color display" is considered slightly more important than "WAP". A consistent answer would be that "WAP" was more important than "Color display".
The more inconsistent answers, like in the example above, the higher the CR is for the prioritization.

The CR from the prioritization above is 0.19, which is well above the recommended limit proposed by Saaty. This example shows that a high inconsistency (i.e. CR > 0.10) is not especially difficult to obtain with only a few variables to prioritize, not to mention if you have e.g. 10 variables (10 variables means 45 prioritizations). The positive thing is that the CR actually indicates consistency, which can be seen as a quality indicator of a person's answers.

The Planning Game is a more informal way in which the improvement issues can be sorted and ordered according to priority. This method is used extensively in extreme programming [27] for feature and release planning, but can easily be used in prioritizing improvement issues.

Ranking involves putting the improvement issues in a list; the higher in the list an issue is the more important it is deemed and *vice versa*. The result of this ranking is a list of ordered issues on an ordinal scale (it is possible to observe that, e.g., issue I is more important than J but not how much more important it is).

The '100-points method' can be used to distribute tokens of some sort (e.g. 'money') amongst the issues. The issues are prioritized with regards to how much 'money' they receive. An example of this could be to give each person doing the prioritization \$100 (or \$1000 etc.) to distribute amongst the improvement issues. The result from this method is a weighted prioritization on a ratio scale (the issues are ordered and a distance between them can be observed, e.g. issue I is \$10 more important than J). For further information see [28].

The primary goal with prioritization is to ascertain what improvement issues are considered important, and which can be put off for future SPI cycles. The choice of technique depends on, for example, how many issues you have to prioritize. AHP demands $n \times (n - 1)/2$ comparisons, e.g. 45 comparisons for 10 issues. This amount is manageable, but when considering 20 issues (190 comparisons) it may start to get unmanageable. To get, say, 10 professionals to sit down and do 190 comparisons (and be consistent) is not easy. Using the 100-point method and giving them the task of distributing \$100 over the 20 issues may be more feasible in this case.

It is important, however, to realize that different prioritization methods have different pros and cons, e.g. AHP gives information about consistency and relative priority, whereas the planning game does not but is much faster in performing when dealing with many improvement issues. A comparison of prioritization techniques is provided in [29].

3.1.4. Weighing the priorities after the fact

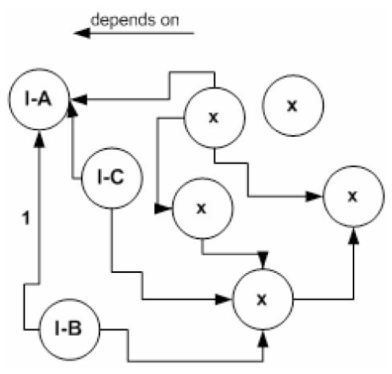
As each contributor's results are gathered there is the possibility to weight the results, and hence enable one to decide on the importance of a certain viewpoint. An example of this could be a desire to premiere developers and their views by multiplying their results by 1.0, while using the multiplier of 0.6 for, e.g., managers, effectively premiering one view over another. This is the same as discussed in Section 3.1.1, where the sample selection was as a weighting method, but this could also be done after the prioritization, independent of sample.

The danger of adding weights is evident, i.e. it could initiate conflicts, as well as belittle valuable views. The possibility of weighting results exists independent of prioritization technique used, but should only be performed after careful consideration, and with clear reasons.

3.2. Mapping dependencies (step B)

This step is aimed at mapping the dependencies between improvement issues. For an example of results obtained during a dependency mapping performed industry, see Section 4.3.3.

This step should involve the same participants as the prioritization (given that the prioritization participants were sampled from the organization in an adequate manner, see Section 3.1.1). Some variations may be necessary, i.e. being able to prioritize issues is not the same as being able to



Dependency example

A real life example of a potential dependency could be if there were two new practices to be introduced into an organization, e.g. *Programming in Java and Object-Oriented Design*. Java is an object-oriented programming language. Therefore the practice of *Object-Oriented Design* should be implemented first (to make use of Java as an object-oriented programming language), i.e. *Programming in Java* is dependent on having an *Object-Oriented Design*. If this example is transferred to Figure 4 *Object-Oriented Design* could be denoted by *I–A* and *Programming in Java* could be denoted by *I–B*. The dependency is denoted by the arrow (1).

The motivation could in this case be 'there is no point in introducing an OO programming language if we cannot design in an OO fashion . . . '.

Figure 4. Dependency diagram example.

identify dependencies between them. Whether or not the prioritization sample can be reused depends largely on the constituents of the sample itself, i.e. which people are involved, where they work (department) and what their abilities are. Identified dependencies are drawn (by each participant) between the issues with two major attributes registered, i.e. *direction* and *motivation*. The arrow itself denotes a dependency between two issues, e.g. *I–B* and *I–A* as illustrated in Figure 4, and the direction of the dependency can be seen through the direction of the arrow. In Figure 4 issue *I–B* depends on *I–A* (as does *I–C*).

In addition to drawing the arrows, a *motivation* for each arrow is registered. This is done to avoid arbitrary and vague dependencies and dependencies can be sorted and compared during the compilation of the results, i.e. it is possible to see if two participants have the same type of dependency between two issues, or if the same arrow denotes two different views. This is also a way in which different types (views) of dependencies can be elicited.

The result of this step should be a list of dependencies between the issues as well as their relative weight, i.e. how many times they are specified by the participants. Table II illustrates the results of an example dependency mapping. The dependencies present (e.g. *I–B* on *I–A*) depend on which dependencies are identified. P1, P2, . . . , and P*n* denote the participants, and the numbers in each participant's row denotes if he/she stated the dependency or not (1 for yes). By summarizing the number of identifications of a certain dependency, weights are ascertained. In this case, *I–B* on *I–A* has a weight of 4, *I–B* on *I–C* has a weight of 3, and so on.

The weights are compiled and drawn into a dependency diagram, see Figure 5, which is like the one presented earlier in Figure 4, but with the addition of weights and 'white' issues (*I–D* and *I–F*) denoting issues that have no dependencies on them. Note that issue *I–F* has no dependencies, and furthermore it is not dependent on any other issue.

If an improvement issue dependency has a low weight, e.g. 1 (*I–C* on *I–E* in Figure 5 has a weight of 1), only *one* person has identified the dependency. Such single occurrences may be a result of misunderstandings and/or other anomalies (given that, e.g., 10 persons participate in the dependency

Table II. Dependency table.

Dependency	P1	P2	P3	P4	P*n*
I–B on *I–A*	1	1	1	1	
I–B on *I–C*	1	1		1	
I–C on *I–A*		1		1	
I–C on *I–E*		1			
I–D on *I–B*	1	1	1		
I–n on *I–n*					

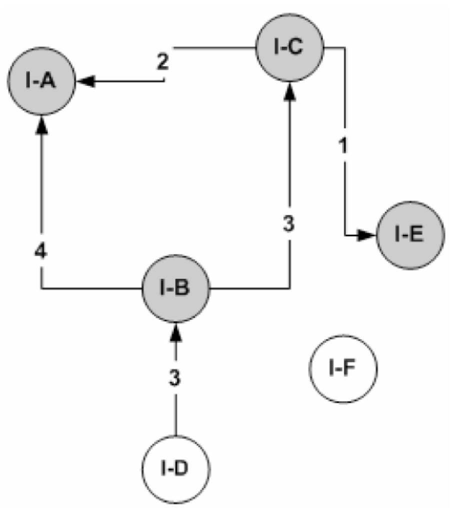


Figure 5. Dependencies with weights.

mapping) and can be omitted to avoid having a great number of weak dependencies to take into consideration. However, all 'anomalies' should be scrutinized by the SPI group before they are dismissed in order to assure that the dependency is not relevant. The SPI group should not dismiss dependencies unless a consensus can be reached about the issue.

A good rule of thumb is that there should be a 'threshold' set by the SPI group beforehand. The idea with mapping dependencies is to catch the important relations between improvement issues. Too many dependencies amongst a large number of issues can result in a dependency diagram that is unmanageable, and thereby useless. On what level this threshold should reside should be governed by the circumstances of the dependency mapping occurrence.

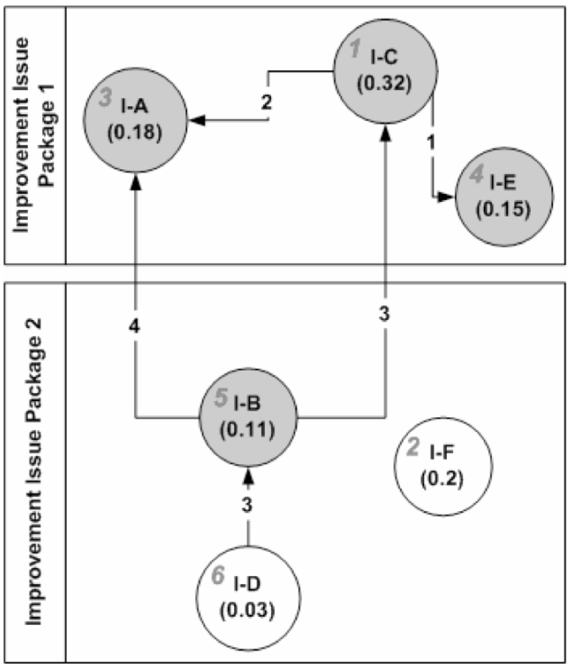


Figure 6. Improvement issue packages.

3.3. Package improvement issues (step C)

Thus far, we have the priority of the improvement issues, and a mapping of the dependencies amongst them. The last stage of DAIIPS is to compile the information in order to generate packages of improvement issues that can be used as a base for planning and implementing an SPI cycle.

For an example of improvement issue packaging in an industry case, see Section 4.3.4.

Figure 6 is a continuation of our example from previous sections with all information gathered in one figure. The improvement issues *I–A* through *I–F* are augmented with relative priority (denoted by numbers within parentheses) and relations with weights. The larger numeral (upper left hand in each circle) denotes the rank of each improvement issue, i.e. *I–C* has the highest priority and *I–D* the lowest (note that all priority methods do not produce relative values of priorities; AHP is used in this example, see Section 3.1.3).

The packaging of the issues largely depends on what is entailed in each improvement issue. In our simple exemplification all issues demand equal time and resources to implement, thus their division into two packages is fairly simple. *I–C* is of highest priority, i.e. it governs that *I–A* and *I–E* be packaged with it due to dependencies on these two issues. Observe that issue *I–F*, which is the second issue in relative priority, is not included in package 1, i.e. the relations have precedence over priority in this example.

Priority, *dependencies* and *cost* (estimated resources demanded for implementation) are the primary variables that have to be adhered to during packaging. The packaging should be done to reflect the current and near-future needs of the organization (*priority*) as well as the available resources for process improvement (attention to *cost*), and last but not least attention should be paid to the order (*dependencies*) of the implementation of improvement issues. It seems that packaging one issue that cannot be realized until another is implemented, e.g. packaging issue *I–A* in package 2 in our example, would be less than optimal. This is not to say that dependencies should always take precedence over the other variables. There are more ways in which the packaging can be performed—the method chosen is up to the SPI group. The main concern in this step is to package the issues so that a compromise between priority, dependencies and cost can be reached as the issues are packaged into units that are appropriate for an SPI cycle. The SPI group decides this with the individual improvement issues as a base, i.e. initial estimation and planning has to be done to ascertain what each improvement issue entails as far as time and resources are concerned. In the example above the division is simple, which may not be the case in reality. Diagrams (like the one displayed in Figure 6) should act as a decision support tool for the SPI group when undertaking the task of establishing what is to be done first, second and so on.

As the three steps are completed the SPI group needs to perform a validity review. This should be an official action performed to (1) ascertain whether the results from the prioritization and dependency mapping are good enough to proceed, and (2) to formally review the documentation produced to ascertain that no omissions/mistakes crept in during the processing of the data gathered from the DAIIPS work. Reviews of dismissed dependencies (e.g. whether the threshold was set at a proper level), and the extent to which high inconsistency (CR) is a threat against the ascertained priority of the improvement issues are examples of important issues. This validity review should help ensure that the quality of the material produced through the usage of DAIIPS is high.

4. INDUSTRY AND ACADEMIA STUDY

This section presents the design and results of the industry and academia (validation) study performed in order to test the DAIIPS framework in an industry setting. The section is structured as follows. In Section 4.2 the designs of the studies are described. The results from the industry study are presented in Sections 4.3.2 (step A), 4.3.3 (step B), and 4.3.4 (step C), corresponding to the three major steps of DAIIPS (see Figure 2). Section 4.4 presents the academia (validation) study results, and in Section 4.5 the industry and academia study are compared in a validation attempt.

The industry study described in this paper is from a SPI project performed at DanaherMotion Särö AB (DHR), where the use of DAIIPS was a part of the SPI work.

DHR develops and sells software and hardware equipment for navigation, control, fleet management and service for Automated Guided Vehicle (AGV) systems. More than 50 AGV system suppliers worldwide are using DHR technologies and know-how together with their own products in effective transport and logistic solutions to various markets worldwide. The headquarters and R&D centre is located in Särö, south of Gothenburg, Sweden. DHR has 85 employees. DHR is certified according to SS-EN ISO 9001:1994 (currently working on certification according to ISO 9001:2000), but there have not been any attempts towards CMM or CMMI certification.

DHR has a wide product portfolio, as the ability to offer partners and customers a wide selection of general variants of hardware and supporting software is regarded as important. Tailoring and especially lighter customer adaptation often follows the procurement and subsequent installation of a system. This in addition to development of new software and hardware makes it a necessity to plan, execute and manage a wide range of projects.

4.1. Related work

The need for continuous process improvement is well known at DHR, where the area of requirements engineering was identified as a good candidate. This paper (the DAIIPS framework and the industry study presented below) is a product of research conducted at DHR based on their need for improvements in their requirements engineering process. Although DAIIPS was formulated in conjunction with requirements engineering process improvement work, it is not tailored towards a single sub-process area (e.g. requirements engineering), but can be used in any process and/or sub-process improvement effort.

A proceeding process assessment concentrated on the area of requirements engineering was conducted at DHR using a lightweight triangulation approach [20].

The nine improvement issues (see Section 4.3) used as input to DAIIPS (as viewed in Table V) came from this assessment of the requirements engineering process at DHR.

4.2. Study design

This section covers the design of the industry study performed. The design is based on the DAIIPS steps described in Section 3, and can be seen as the preparation for the use of DAIIPS.

In addition to performing a study in industry, a second study was performed in academia. The purpose of the academia study was to validate some of the results obtained in the industry study, e.g. an effort to secure the external validity of the study (see Section 4.2.3.3), and to help in confirming that no important dependencies were missed during the industry study, i.e. the use of DAIIPS in an industry SPI effort. An overview of the studies is illustrated in Figure 7. Improvement issues (obtained during process assessment) are used as input for DAIIPS used in the two studies. Observe that only the dependency mapping was performed during the academia study, as the prioritization results (see Section 4.4) were not used.

4.2.1. Industry study design (preparation step A and B)

4.2.1.1. Prioritization preparation (step A). **Sampling** (see Section 3.1.1) of the subjects for the prioritization (and later the dependency mapping) was based on the sample selected for the process assessment (PA) conducted (as mentioned in Section 4.1). It was based on quota sampling, in an effort to get subjects from different parts of the population [21]. The quotas were selected based on ten different roles identified in development at DHR.

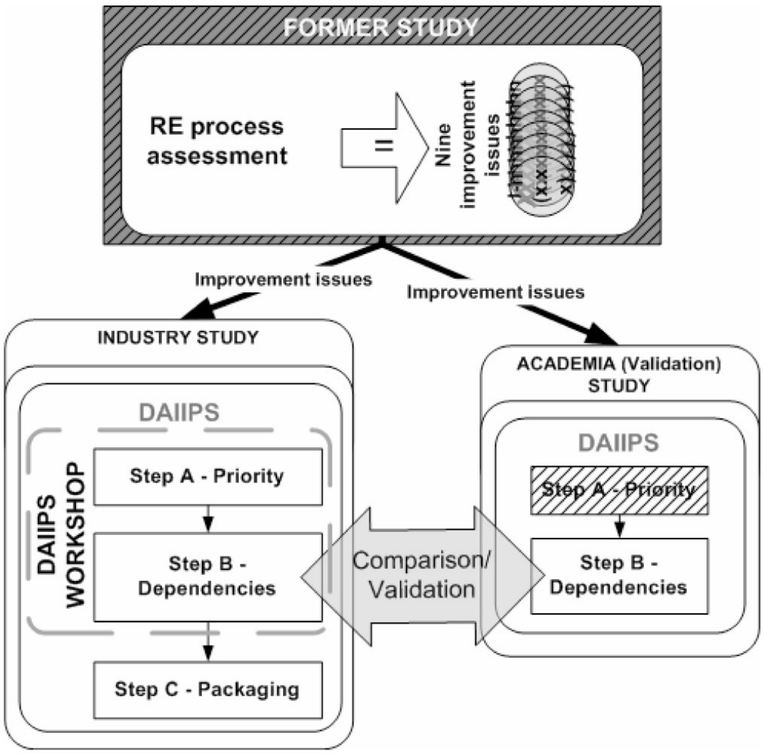


Figure 7. Study overview.

1. The *Orderer* has the task of being the internal owner of a certain project, i.e. has the customer role and if applicable the official contact with an external customer and/or partner. This party is responsible for the official signing-off when it comes to the requirements, i.e. he/she places an order.
2. The *Project Manager* has the traditional role of managing the project, resources, planning and follow-up. As far as requirements are concerned the Project Manager is responsible for ensuring that the requirements engineering is performed, the requirements specification is written and signed off by the System Engineer.
3. The *System Engineer* is technically responsible for a project. It is also important to recognize that the System Engineer has the official responsibility for the requirements specification in a project.
4. The *Developer* is a representative of the developers (e.g. programmers) in a project, those actually implementing the requirements. The developers use the requirements specification.

Table III. Participant distribution over roles—prioritization.

Role	No. participants	Role	No. participants
1	1	6	1
2	2	7	1
3	2	8	2
4	2	9	1
5	1	10	1

5. *The System Test* role can be described as the traditional role of application and feature testing. This role is officially present during initial project meetings and is a part of the verification of the requirements specification.
6. *The Production* role also has a presence in projects. This representation consists of verifying that the production aspects are met, i.e. that production is taken into consideration at an early stage.
7. *The Application Developer* role represents installation and adaptation aimed at industry partners and/or end customers. Adaptation here translates to tailoring, development of some light customer features and some support.

In addition to having representatives for the roles presented above three more general roles were represented.

8. *Management*
9. *Sales/Marketing*
10. *Sales Support*

The last three roles were more general in nature in comparison to the development-specific roles presented above. In total 14 people participated, distributed over the roles as can be viewed in Table III.

The reason for having ten roles and 14 participants (i.e. an overrepresentation of some roles) was twofold. First the overrepresented roles housed most senior personnel and it was deemed positive to maximize the amount of senior staff participating. The second reason was executive, i.e. the roles of Project Managers and System Engineers to a degree represented the (in this study) undefined role of middle management.

Criteria (see Section 3.1.2) were based on maximizing the quality of the produced products in terms of customer satisfaction. The focus was to increase the quality of the requirements engineering process, thus meeting the objective of increasing customer satisfaction.

Prioritization technique (see Section 3.1.3) was chosen because there were only nine improvement issues to prioritize. With this in mind AHP was suitable, and could provide priority and information about, for example, consistency (see Section 3.1.3—AHP for further information, and [29] for an evaluation of AHP in comparison to other prioritization techniques).

Table IV. Participation distribution over roles—dependency mapping.

Role	No. participants
1: Orderer	1
2: Project Manager	2
3: System Engineer	1
4: Developer	2
5: System Test	1
6: Production	*0*
7: Application Developer	1
8: Management	2
9: Sales/Marketing	*0*
10: Sales Support	*0*
Total	10

4.2.1.2. Dependency mapping preparation (step B). **Dependencies** were mapped by the same group of professionals as described in the sample above (participants in the same DAIIPS workshop) but with a modified representation over the different roles. In total there were ten participants, as can be seen in Table IV.

Some roles, i.e. Production, Sales/Marketing and Sales Support, were not present during the dependency mapping, while other roles more directly linked to system development (active participants and 'owners' of the process) were represented.

The main reason for not eliciting all roles was that the dependencies between the improvement issues were not obvious to people not working directly with the development, and thus the input from these roles was not premiered. The ten persons chosen for the task of dependency mapping were all active participants within the research and development department, many of which were senior members of staff with experience from multiple roles over a number of years.

The roles not elicited during the dependency mapping were however present during the prioritization. The rationale behind the two samples was that all roles (presented in Section 4.2.1) could have relevant and important input as to what parts of a process needed to be improved. Knowledge of how the improvement issues were dependent on each other, however, was deemed better explored by those that worked with the process every day.

4.2.2. Academia (validation) study design

Sampling (see Section 3.1.1) of the subjects for the prioritization (and later the dependency mapping) was based on convenience sampling [22]. The sample consisted of six PhD students from the Department of Software Engineering & Computer Science at Blekinge Institute of Technology. The students in question were two senior students (had been PhD students for more than 2 years), and four junior students (<2 years).

Criteria were set based on the criterion used in the industry study, i.e. maximizing the quality of the RE process.

Prioritization technique and **dependency mapping** were done in the same manner as in the industry study described in Section 4.2.1.

4.2.3. Validity evaluation

In this section we discuss the threats to this investigation. We base this on the discussion of validity and threats to research projects presented in Wohlin *et al.* [21]. One type of threat mentioned in [21] is not relevant, since the investigation was conducted in an industrial environment. The threat not considered is construct validity, which is mainly concerned with the mapping from the real world to the laboratory. The investigation presented here, however, was conducted in the real world. The validity threats considered are conclusion, internal and external validity threats.

4.2.3.1. Conclusion validity. The questionnaire used for the prioritization and dependency mapping was validated through preliminary testing and proofreading by several independent parties to avoid factors like poor question wording and erroneous formulation.

Each prioritization and dependency mapping was done in one uninterrupted work session. Thus the answers were not influenced by internal discussions about the questions during, e.g., coffee breaks.

The sampling techniques used for the industry study could have posed a threat to the validity of the investigation. The subjects selected may not have been totally representative of the roles at DHR. The main assurance that this misrepresentation is minimal is the fact that the subjects were selected in cooperation with three senior managers with extensive knowledge and experience with regards to the development processes and the personnel at DHR.

4.2.3.2. Internal validity. As the prioritization and dependency mapping was done on paper (i.e. there was a record of people's opinions and views) this could have constrained people in their answers. This potential problem was alleviated by the guarantee of anonymity to all participants. Thus all information divulged during the study was only to be used by the researchers.

4.2.3.3. External validity. The external validity is concerned with the ability to generalize the results. As far as the results pertaining to priority are concerned this is not a significant threat to the study, since the objective is not generalizing DHR's priorities to other environments (i.e. things important to DHR may be less critical for another company). The important generalization here is whether the applied approach for prioritizing, dependency mapping and packaging improvement issues is applicable to other environments. There is nothing in the approach that makes it tailored to a specific setting, hence the approach should be useful at other SMES in choosing what improvement issues to undertake in their SPI enterprise.

The academia study was performed to validate that the identified dependencies between the improvement issues were state-of-the art, i.e. that no important and unforeseen dependencies were missed in the industry study. As far as the priority of improvement issues obtained from the academia study is concerned no real validation and/or comparison is relevant. This is because the PhD students are not employees of the DHR organization, thus they have no appreciation or insight into the strengths or weaknesses perceived by DHR employees.

Table V. Improvement issues at DHR.

Improvement issues (in no specific order)
Issue 1: Abstraction level and contents of requirements *Each requirement should be specified on a predefined level of abstraction with certain characteristics (attributes attached to it), enabling requirements to be comparable and specified to a certain predefined degree of detail.*
Issue 2: Requirements prioritization *This issue suggests a systematic prioritization of all requirements in a standardized way.*
Issue 3: Requirements upkeep during and post project *In order to keep the requirements up to date during and post project the requirements have to be updated as they change.*
Issue 4: Roles and responsibilities—RE process *To avoid misunderstandings as well as avoiding certain tasks not being completed the roles and responsibilities of all project members should be clearly defined before project start.*
Issue 5: System tests performed against requirements *All system tests should be performed against requirements (e.g. using requirements as a base to construct test cases).*
Issue 6: RE process/methods *This issue is basically the creation and incorporation of a complete and comprehensive RE process at DHR that is well defined and documented.*
Issue 7: Requirements reuse *By reusing requirements everything from the requirement itself to analysis, design, implemented components, test cases, scenarios, and use cases etc. can be reused.*
Issue 8: Requirements traceability *Policies and support for traceability to and from the requirements are to be established.*
Issue 9: Requirements version handling *Policies and support for version handling of each requirement (not only on requirement's document level) should be established.*

4.3. Industry study execution and results

The first part of the DAIIPS workshop (see Figure 7 and Section 3) was to introduce the participants to the nine improvement issues and the official formulation of these issues, i.e. what each issue means and what they involve. This was a straightforward procedure since all participants had taken part in the PA activities earlier. The improvement issues can be viewed in Table V. The descriptions of the issues (in italic under each issue) are given in abbreviated format.

The next part was to initiate the workshop. A general description covering the form, how answers should be specified, and information about anonymity issues preceded the handing out of the forms. Examples were given as to the meaning of the priority scale and questions were answered.

Table VI. CRs for DHR participants.

Subject	CR	Subject	CR
DHR6	0.05	DHR1	0.17
DHR9	0.11	DHR5	0.19
DHR7	0.12	*DHR12*	*0.28*
DHR4	0.12	*DHR13*	*0.30*
DHR14	0.13	*DHR8*	*0.35*
DHR2	0.14	*DHR10*	*0.38*
DHR3	0.14	*DHR11*	*0.59*

The organizing body of this workshop was present during the entire duration and the participants could ask questions as needed throughout the prioritization.

The handout included several examples of how prioritization and dependencies were to be specified. The workshop lasted for over 2 h.

4.3.1. Sample and consistency ratio

During the compilation of the prioritization results the consistency ratio (CR) was observed for each participant. It varied from a CR of 0.05 (well below the recommended limit) to 0.59 (regarded highly inconsistent). The average CR was ≈0.22 (median ≈0.16). A summary of the participants, CRs is given in Table VI.

The decision was made to include only the results from participants having a CR of 0.20 or less. Going by the recommendation by Saaty of 0.10 would have excluded all results except for one participant, therefore this would not have been a practical solution.

This of course had a negative impact on the sample, i.e. the distribution of participants over the roles was altered due to the invalidation of some results (see Table VI, results in italic).

In Table VII the impact of the participants with too high CRs (>0.20) can be observed. In the left column the roles are listed and the middle column displays the number of people that participated in the prioritization for each role. The last column denotes the total number that was incorporated in the prioritization results, i.e. those with a CR of less than 0.20 (numbers is parentheses represent the number of people that were excluded from the prioritization). The bottom row shows the total, i.e. five people were excluded, leaving nine people.

Some roles were diminished and others vanished altogether, i.e. the Production and Application Developer roles were not to influence the prioritization of the improvement issues at all. This ‘total non-representation’ by the two roles was however deemed acceptable, as was the diminished representation of three other roles. The reason for this was that the priority of the improvement issue differed relatively little after excluding the persons with high CRs (see Section 4.3.2 and Figure 1).

Table VII. Result of invalidation impact on sample.

Role	No. participants	Result (difference)
1: Orderer	1	1 (0)
2: Project Manager	2	2 (0)
3: System Engineer	2	1 (−1)
4: Developer	2	1 (−1)
5: System Test	1	1 (0)
6: Production	1	0 (−1)
7: Application Developer	1	0 (−1)
8: Management	2	1 (−1)
9: Sales/Marketing	1	1 (0)
10: Sales Support	1	1 (0)
Total	14	9 (−5)

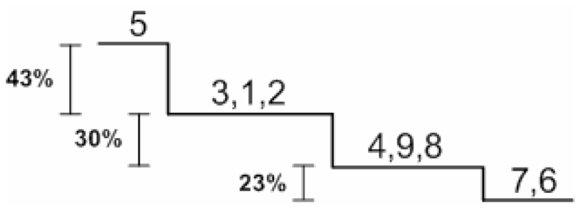


Figure 8. Priority shelves.

4.3.2. *Priority of improvement issues (results—step A)*

The results of the prioritization can be viewed in Table VIII, in which all nine issues are present. The views of the participants are fairly scattered, although some tendencies of consensus can be observed.

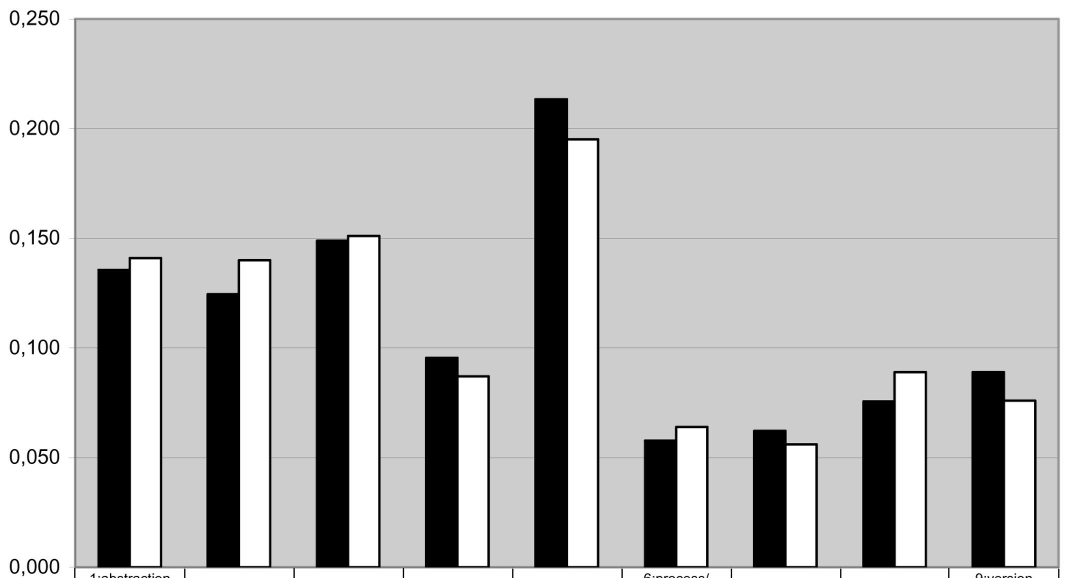
From the *rank* row in Table VIII it can be seen that issue 5: System Test has the highest priority by a margin of almost 43% ((0.213 − 0.149)/0.149) in comparison to issue 2: Upkeep ranked as no. 2.

The issues ranked second to fourth only have a moderate difference in priority in comparison (i.e. the difference between issues 2 and 4 is a moderate 20% ((0.149 − 0.124)/0.124)).

Looking at Figure 8 the issues are grouped together on a 'shelf structure' which illustrates the jumps in priority. The top shelf is occupied with issue 5, then there is a large drop in priority to the next shelf where issues 3: Requirements upkeep during and post project, 1: Abstraction level and contents of requirements and 2: Requirements prioritization are grouped, and so on. It is noticeable that there are several jumps in priority, and there is a substantial difference between the issues ranked in the top four and the issues ranked five and lower.

Table VIII. Priority results (CR < 0.20).

ID	1: Abstraction/ contents	2: Prioritization	3: Upkeep	4: Roles	5: System test	6: Process/ methods	7: Reuse	8: Traceability	9: Version handling	CR
DHR1	0.22	0.07	0.19	0.03	0.26	0.03	0.05	0.07	0.09	0.17
DHR2	0.2	0.14	0.19	0.05	0.22	0.07	0.06	0.06	0.02	0.14
DHR3	0.17	0.07	0.18	0.28	0.1	0.02	0.04	0.05	0.09	0.14
DHR4	0.08	0.24	0.08	0.09	0.25	0.12	0.06	0.03	0.04	0.12
DHR5	0.12	0.14	0.14	0.03	0.17	0.05	0.06	0.1	0.2	0.19
DHR6	0.04	0.16	0.17	0.17	0.05	0.12	0.09	0.13	0.07	0.05
DHR7	0.13	0.15	0.19	0.05	0.18	0.03	0.1	0.08	0.1	0.12
DHR9	0.14	0.09	0.14	0.06	0.25	0.03	0.04	0.11	0.14	0.11
DHR14	0.12	0.06	0.06	0.1	0.44	0.05	0.06	0.05	0.05	0.13
Average (C < 0.20)	0.136	0.124	0.149	0.096	0.213	0.058	0.062	0.076	0.089	0.130
Rank	3	4	2	5	1	9	8	7	6	



	1:abstraction /contents	2:prioritization	3:upkeep	4:roles	5:system test	6:process/ methods	7:reuse	8:traceability	9:version handling
■ CR < 0.20	0,136	0,124	0,149	0,096	0,213	0,058	0,062	0,076	0,089
□ CR > 0.20	0,141	0,14	0,151	0,087	0,195	0,064	0,056	0,089	0,076

Figure 9. Priority comparison.

There is quite a large difference in opinion regarding issue 5: System test (ranked number one), i.e. a scattering from a priority of only 0.05 to an extreme 0.44 (i.e. more than eight times as high). The scattering is less in issues 2–4 which lie closer in priority.

Figure 9 also illustrates the priority of the improvement issues. There are two series of data presented, the priorities of the issues by only the participants with CR < 0.20 (black bars), and the priorities of all 14 participants, i.e. not excluding those with CR > 0.20 (white bars). It can be seen that the priority of the improvement issues is not substantially changed when the results of all 14 participants are taken into consideration, at least not for the first two shelves, i.e. issue 5 is still on step 1, and issues 3, 1, 2 are still on step two. The changes amongst the other issues are also rather moderate.

4.3.3. Dependencies between improvement issues (results—step B)

As the results from the dependency mapping were compiled, the weight of each dependency was recalculated in per cent (i.e. what percentage of the subjects identified the dependency), and a threshold was set at 20%, i.e. more than 20% of the subjects doing the dependency mapping had to identify the dependency (see Section 3.2).

Table IX. Dependencies between improvement issues identified at DHR.

Dependency (issue i on issue j)	Weight (%)	Dependency (issue i on issue j)	Weight (%)
2 on 1	60	5 on 8	20
3 on 1	*20*	7 on 1	30
3 on 4	40	7 on 3	60
3 on 6	*20*	*7 on 6*	*20*
3 on 9	*20*	7 on 8	40
4 on 6	*20*	7 on 9	30
5 on 1	60	8 on 1	20
5 on 2	30	*8 on 3*	*10*
5 on 3	70	*9 on 3*	*10*

The results from the dependency mapping can be seen in Table IX, where dependencies deemed to be under the threshold are given in italic. The dependencies under the threshold were scrutinized and dismissed, all except one, i.e. dependency *8 on 1* (8: traceability was deemed dependent on 1: abstraction/contents). The reasoning behind this was discussed and consensus reached thus making an exception to the general threshold limit.

A dependency diagram is given in Figure 10. Here the relations, their weight (e.g. 0.6 is the same as 60% in Table IX), and the relations' direction can be observed. Relative priority (the value inside the parentheses) and rank (top bold numeral) are also given.

4.3.4. Package improvement issues (results—step C)

Issue 5: System test has the highest priority by far, and depends on issues 1, 2 and 3. Issue 3 in turn depends on issue 4. This 'unit', i.e. issues 5, 3, 2, 1 and 4, was broken out and put into an SPI package, see Figure 11. This was a seemingly optimal choice with regards to priority and the mapped dependencies. However this packaging was rejected due to the strain on resources such a large SPI package would demand (especially in terms of time to return on investment). Instead a second proposal for packaging was drawn up taking resources and time into account.

The previously suggested SPI Package A (see Figure 11) was divided into two packages, i.e. 2: Prioritization and 5: System test was broken out to its own package (see Figure 12). This meant that issue 5 (which had the highest priority) was postponed. The reasoning for this was that SPI Package 1 (see Figure 12) was a prerequisite for SPI Package 2, and it would probably be implemented in this order anyway, i.e. even if the packaging had been accepted from the first suggestion as seen in Figure 11. The major difference in breaking up package A further was for the implementation and evaluation of the implementation to be faster. If package A had been kept according to the original suggestion more resources and time would have been required before feedback on the implementation of the activities was available. By dividing the package an evaluation of the work could be done sooner, and a subsequent decision to continue (i.e. with package 2 in Figure 12) or not could be made earlier and reduced costs.

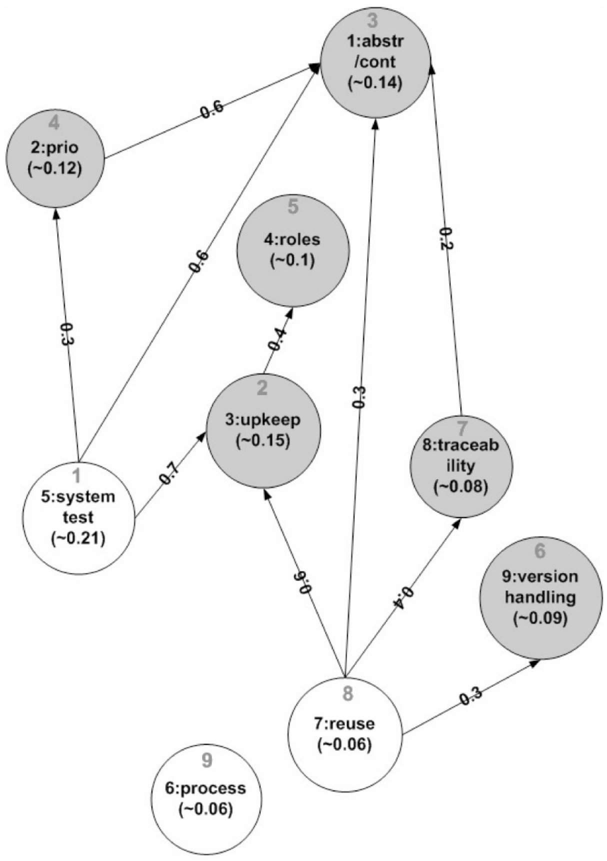


Figure 10. DHR dependency diagram.

4.4. Academia (validation) study execution and results

The study in academia preceded much in the same manner as the industry study. The subjects selected were given the same information and the same form to fill out regarding prioritization and dependency mapping of improvement issues.

The workshop was expected to take 2 h, but only took about an hour in total.

4.4.1. Sample and consistency ratio

During the compilation of the prioritization results the CR was observed for each participant. It varied from 0.23 (well above the recommended limit set by Saaty and the limit deemed acceptable during the industry study) to 0.62 (regarded as highly inconsistent). The average CR was ≈0.40 (median ≈0.35). A summary of the participants, CRs can be viewed in Table X.

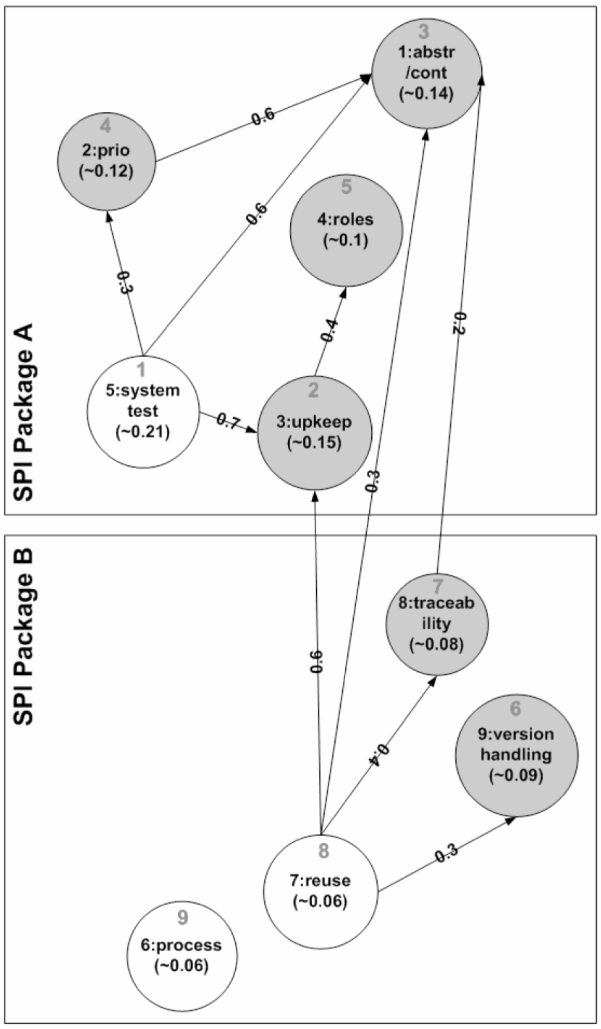


Figure 11. SPI package suggestion at DHR.

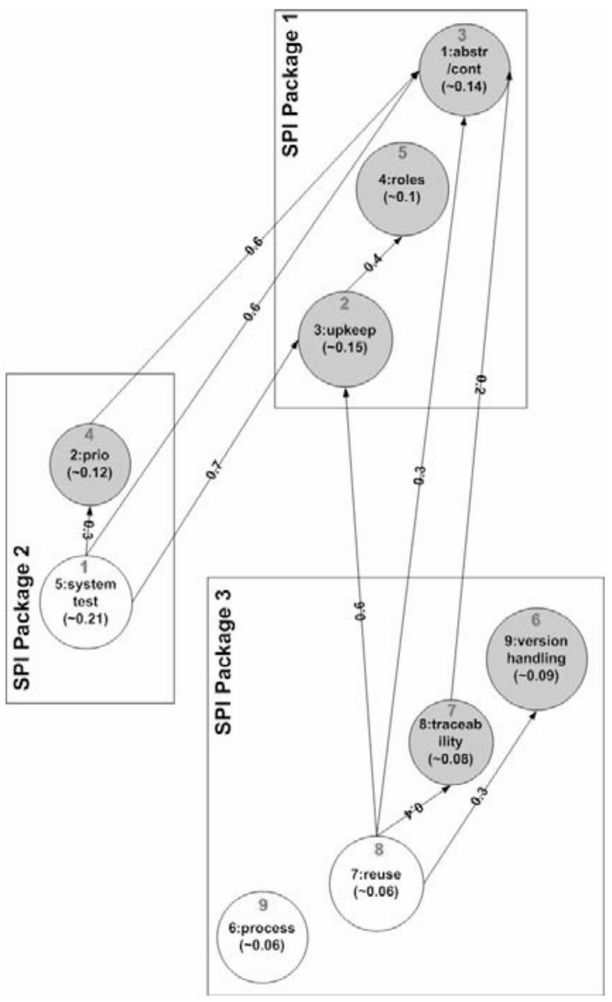


Figure 12. Final SPI package at DHR.

As stated in Section 4.2.3.3, a comparison between academia and industry as to the priority of improvement issues was not considered relevant. The CRs were very high over all (well over the limit of CR < 0.2 used in the industry study), in general disqualifying all participants in the academia study. However, this is of subordinate importance as the academia study was not done in order to confirm/validate the priorities obtained from industry, but rather to compare the dependencies being mapped in order to get an idea if the dependencies found in the industry study were also found in academia, and *vice versa*.

Table X. CRs for academia participants.

Subject	CR
BTH1	0.42
BTH2	0.62
BTH3	0.27
BTH4	0.23
BTH5	0.26
BTH6	0.58

Table XI. Dependencies between improvement issues identified in academia.

Dependency (issue *i* on issue *j*)	Weight (%)	Dependency (issue *i* on issue *j*)	Weight (%)
2 on 1	83	6 on 4	33
3 on 1	33	7 on 1	67
3 on 4	33	7 on 3	33
3 on 6	50	7 on 6	33
3 on 9	33	7 on 8	33
4 on 6	33	7 on 9	50
5 on 1	83	8 on 1	33
5 on 3	83	8 on 3	50
5 on 8	50	9 on 3	33

4.4.2. Dependencies between improvement issues

As the results from the dependency mapping were compiled, the weight of each dependency was recalculated in per cent (what percentage of the subjects identified the dependency), and a threshold was set at 20%, i.e. more than 20% of the subjects doing the dependency mapping had to identify the dependency (see Section 3.2).

The results from the dependency mapping can be seen in Table XI.

It is noticeable that no dependencies identified were specified by less than 33% (i.e. two participants in this case).

4.5. Comparison—industry vs. academia

4.5.1. Introduction

In this section some comparisons are made between the dependencies identified in the industry study at DHR and the study performed in academia at Blekinge Institute of Technology. The motivation for

Table XII. Dependencies comparison between DHR and academia.

Dependency (issue i on issue j)	Weight (%) DHR	Weight (%) academia	Note (group)
2 on 1	60	83	1
3 on 1	20	33	4
3 on 4	40	33	2
3 on 6	20	50	3
3 on 9	20	33	4
4 on 6	20	33	4
5 on 1	60	83	1
5 on 2	30	*0*	6
5 on 3	70	83	1
5 on 8	20	50	3
6 on 4	*0*	33	5
7 on 1	30	67	2
7 on 3	60	33	2
7 on 6	20	33	4
7 on 8	40	33	2
7 on 9	30	50	2
8 on 1	20	33	2
8 on 3	10	50	3
9 on 3	10	33	5

the comparison is to validate the dependencies identified in industry (see Section 4.2.3.3). In addition, this section highlights the differences in outlook on dependencies between industry and academia.

Table XII presents a summation of the dependencies presented previously (i.e. in Tables IX and XI). The weights for both industry (column two) and academia (column three) are presented an percentages. The dependencies are also presented in Figure 13 where dependency weights are presented on the lines (industry | academia).

4.5.2. *Comparison analysis*

Below an analysis and augmentation is provided on the dependencies listed in Table XII. The note column indicates numbers that have corresponding numbers in the text below, i.e. some analysis is offered for each number. In some instances the dependencies are grouped together with regards to common properties or tendencies.

Note 1. There is strong support for the dependencies in this group, both in industry and in academia. All of the dependencies in this category were taken into consideration during the packaging activity performed (see Section 4.3.3). The dependencies here are not analyzed or commented upon any further.

Note 2. This group also holds dependencies that were all taken into consideration during the dependency mapping, i.e. all had a weight above the threshold in both industry and in academia. However, in comparison to group 1, the weights are lower, i.e. most weights are $\leq$50%. In addition

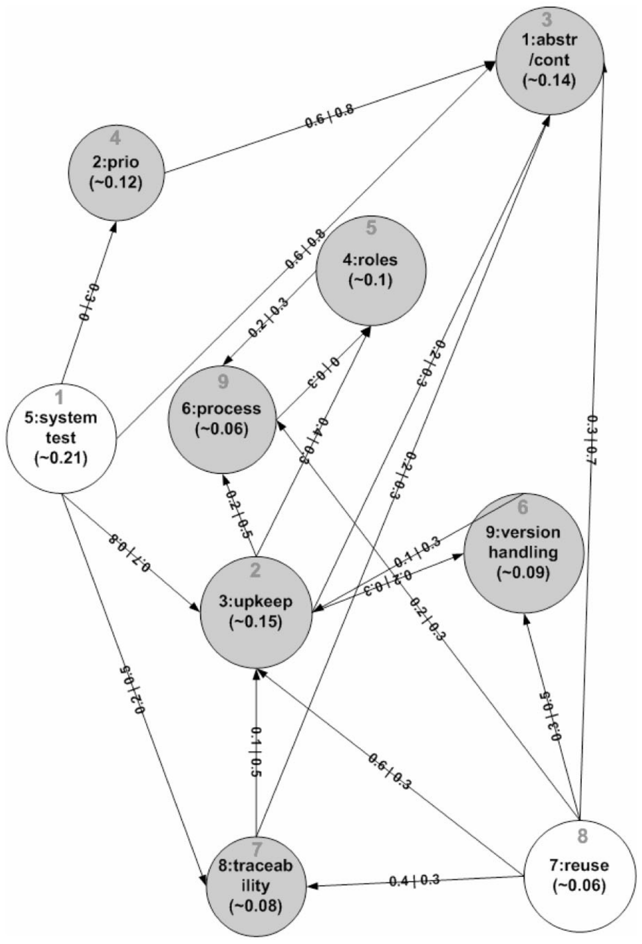


Figure 13. Dependency diagram industry and academia (DHR priority | academia priority).

to this there are some discrepancies between industry and academia. This is especially clear in the case of dependency *7 on 1* (having a much stronger weight in academia than in industry), and *7 on 3* (showing the opposite, i.e. stronger weight in industry). In academia, issue 7: Requirements reuse is considered to be linked primarily to how the requirements are specified (issue 1) while the dependency on issue 3: Requirements upkeep during and post project is considered of less weight. In industry the tables are turned, i.e. issue 3 is premiered. One explanation for this could be that the participants from industry see out-of-date documents, e.g. requirement specifications, as an undesirable but relatively common occurrence in projects, while people based in academia in this case may have premiered other aspects as they are not faced with this particular problem in their daily work.

Note 3. In this group there is a clear discrepancy between industry and academia, i.e. dependencies identified as rather strong (50%) in academia are considered weak in industry ($\leq$20%). This group has the strongest candidates for adding to the dependencies identified in industry, i.e. catching dependencies missed in the industry mapping. The three cases in this group are analyzed one by one below.

3 on 6. In the case of dependency *3 on 6* (6: RE process/methods) the motivation behind this in academia was generally that a documented process helps in the upkeep of requirements. In industry, however, the general perception was that the presence of a documented process does not ensure anything, and that focus should be put on establishing practices for issues, and that a complete and comprehensive documented process is not a prerequisite for any of the issues identified.

Looking at issue 6: RE process/methods there are a number of instances where dependencies are identified both to and from this issue (in groups 4 and 5). In industry the identified dependencies were under the threshold (see Section 4.3.3) in all cases and thus had no impact during the dependency mapping. In academia the dependencies were over the threshold. One possible explanation for industry not premiering this issue (either in the case of prioritization or during the dependency mapping) could be that there is some disillusionment towards official documented processes, i.e. in many cases representing a largely unused set of steps, rules, practices etc. The mindset was that documenting something and making it official does not necessarily promote it or make it useful to the participants of the organization.

5 on 8. In the case of dependency *5 on 8* the general reasoning in academia was that there had to be traceability policies in place in order for system tests to be performed against requirements. This would enable, for example, faster tracking of faults from test cases (based on requirements) and the modules/classes/code.

A dependency between test and traceability was also identified in industry but from a different perspective, i.e. the reason given was that the test could be performed on requirements only if it were possible to trace which requirements were implemented. This rationale (and thus the dependency) was not seen as critical, however, as all requirements allocated to a certain project should be implemented. If the requirements in question were not to be implemented for any reason they would be removed from the project.

In the case of this dependency there is a good chance that the academia view could benefit the industry case.

8 on 3. The reasoning behind this dependency in academia was that requirements not updated were prone to make traceability difficult, if not impossible. However, this dependency was not considered critical in industry.

In the case of this dependency there is a good chance that the academia view could benefit the industry case.

Note 4. In this group the dependencies were identified in both cases, but the weight in industry was just below (or actually on) the threshold, i.e. 20%. In the case of academia all the dependencies had a weight of 33%, just above the threshold, i.e. identified by two persons (two out of six = 33%). There are some implications that could be attributed to this, i.e. that one person less in the case of academia would make this category a non-issue in this analysis.

Given that the dependencies here are rather weak, looking at the combination of both cases the dependencies in this group are not considered as strong candidates for adding to the dependencies identified in industry. This is further supported by the fact that two out of the four dependencies in this group are on issue 6: RE process/methods (the rationale behind dependencies and this issue was discussed above under note 3).

The two other dependencies under this group are discussed below.

3 on 1. The reasoning behind this dependency in academia was that the abstraction level and contents of a requirement made it easy or hard to keep a requirement updated. This dependency was also identified in industry, but the reasoning was rather that someone had to keep the requirements updated (thus the dependency on issue 4), i.e. identifying *how* the requirement was specified could impact on keeping it updated.

3 on 9. This dependency was identified in both studies, and the rationale behind it was that version handling helped in the upkeep of requirements. However, this was not considered critical in industry because version handling of the requirements document was thought to be adequate for the time being and furthermore issue 9 would be implemented later, possibly in combination with CASE tool support.

Note 5. This group shows a large discrepancy between industry (dependency weights well below the threshold) and academia (dependency weights just below the threshold). The weights are fairly weak even in the academia case, thus these dependencies are not considered as strong candidates for adding to the dependencies identified in industry. This is further supported by the fact that one out of the two dependencies in this group is from issue 6: RE process/methods (the rationale behind dependencies and this issue was discussed above under note 3).

In the case of dependency *9 on 3* the reasoning in academia was that upkeep of requirements was the basis for creating versions, i.e. the point of creating versions.

In industry the dependency was seen as versions that had to be created manually (i.e. up-kept). This reasoning was not seen as critical since the idea was that this would be handled by some sort of tool later on (e.g. a CASE tool as mentioned in relation to dependency *3 on 9*).

Note 6. The dependency 5: System tests performed against requirements on 2: Requirements prioritization was identified in industry but not in academia. The motivation behind this dependency in industry was that in order to know what requirements to premiere in the testing it was important to ascertain which were prioritized, implying that there were not always the resources available to pay equal attention to all parts and test all parts as extensively as would be preferred.

4.5.3. Comparison summary and discussion

It should be restated that the comparison between dependency mappings in industry and academia was performed to validate the dependencies found in industry, i.e. strengthening the external validity of this study by alleviating the risk of crucial dependencies being missed.

Two prime candidates for adding to the dependencies found in industry were identified through the academia study, namely *5 on 8* and *8 on 3* (see note 3).

The impact of possibly adding these dependencies to the industry case has to be analyzed.

In the case of dependency *8 on 3* the effects on the SPI package are minor, almost non-existent. Looking at Figure 12, which displays the final SPI packaging at DHR, no action has to be taken, i.e. the new dependency does not force any reallocation of improvement issues within packages 1–3.

In the case of dependency *5 on 8* the impact is greater. In order for the dependency to be satisfied issue 8 has to be moved to either package 2 (and supersede implementation of issue 5), or package 1.

In either case there is a reallocation of an improvement issue and a subsequent increase in the size of the affected package.

Looking at the other dependency comparisons the discrepancies between the studies were less significant. Most dependencies identified in one study were also seen in the other, with a few exceptions.

Some of the most prominent discrepancies were related to dependencies to and from issue 6: RE process/methods. The industry view was more pragmatic than that of academia, i.e. recognizing that an official and documented process is seldom a guarantee for things actually being done, not to mention done in a certain way which is set down on paper.

This difference in outlook between industry and academia is also seen in the case of testing being dependent on having prioritization of requirements, i.e. *5 on 2* (note 6). The assumption in academia is that all implemented components are tested. This is true in the industry case as well (as probably all parts are tested), but limited resources and tight deadlines may result in some prioritization as to the degree of testing.

5. DISCUSSION AND CONCLUSIONS

In the introduction to this paper several reasons were given motivating the construction of the DAIIPS scheme. The primary motivation was to give SMEs a decision support scheme that considered several aspects identified as critical for the success of SPI endeavors. The need for this was initially identified in industry, i.e. in the case of the SPI activity performed at DHR.

In the first part (Section 3) we presented an overview of DAIIPS, how improvement issues are prioritized, and how dependencies between issues are identified and visualized. The objective is that this information is used as input to make an informed decision regarding what actions to take in terms of process improvement.

The developed method was subsequently used in an industrial setting (Section 4) to allocate improvement issues into SPI packages, taking priority, dependencies and (to a certain extent) cost into consideration.

Modularity was also premiered in the development of DAIIPS. The scheme is independent, so that any method (e.g. CMMI, SPICE etc.) can be used for the preceding SPA activity, as long as the assessment data are available. The same is true for the implementation of the issues after the DAIIPS packaging, enabling organizations to use DAIIPS as a decision support scheme regardless of the environment. This is of course dependent on certain factors, for example, these must be input from the SPA in some comparable form, i.e. considering abstraction level.

Looking at the industry case presented in this paper, the focus was not on general SPI but rather on improving a sub-process, namely requirements engineering. In the DHR case this was the area that needed to be improved—using DAIIPS enabled the creation of decision support material for the continuation of the improvement work after the assessment stage.

SPI efforts are often an expensive undertaking, thus effectively raising the threshold for companies and especially SMEs. The general idea behind DAIIPS is to offer a way in which the threshold can be lowered, by offering a structured way to increase the control of factors such as *time* to return on investment and *cost* of SPI.

Furthermore, other points argued as critical for the success of SPI undertakings (see Section 2.1) were also taken into consideration during the development and use of DAIIPS.

- *Commitment* by management and middle management is easier to secure if management feels in control of the issues of *cost* and *time*.
- *Commitment* by staff, e.g. engineers (often seen as the most critical issue contributing to SPI success), can be positively influenced by the fact that the prioritization and dependency mapping is largely done by representatives from their 'ranks'.
- *Focus* on a delimited number of improvement issues at a time (i.e. a SPI package) offers clear and well-defined goals that are obtainable.
- *Involvement* (in the SPI work by staff) is easier to secure if the staff have a say in what is to be done from the start.

Enabling the professionals in the SPI-targeted organization to 'take matters into their own hands', prioritizing, mapping dependencies and packaging issues according to their needs, should be premiered. It is important to remember that no method, framework or scheme (or even DAIIPS for that matter) is useful if the professionals, whose organization is targeted for SPI, are not committed.

In summary, a method for prioritization and identification of dependencies between SPI proposals has been presented. The method was successfully applied in an industrial case study involving an organization classified as a small- to medium-sized company.

6. FURTHER WORK

Refinement, expansion and replication are three key words that are central for the future of the DAIIPS scheme.

DAIIPS as a scheme needs to be refined through tweaking of the steps of which it is composed. Lessons learned from the studies thus far need to be evaluated, weaknesses need to be identified and dealt with by further simplification of the scheme where possible.

DAIIPS could be augmented by the incorporation of cost as an official and modeled part of the scheme, thus offering further dimensions to the decision support offered.

Replication of the study presented above is almost a prerequisite in order for the DAIIPS scheme to be tested, and in doing so producing results that allow DAIIPS to evolve. This includes testing the use of alternative prioritization techniques.

ACKNOWLEDGEMENTS

The work and results presented in this paper were done and produced in close and fruitful cooperation between research and industry. The authors would like to thank everyone involved during the SPI work at DHR, and especially the members of the SPI work team for their commitment and tireless efforts.

This work was partly funded by The Knowledge Foundation in Sweden under a research grant for the project 'Blekinge—Engineering Software Qualities (BESQ)' (http://www.ipd.bth.se/besq).

REFERENCES

1. Sommerville I. *Software Engineering* (6th edn). Addison-Wesley: London, 2001.
2. Calvo-Manzano Villalón JA, Cuevas Agustín G, San Feliu Gilabert T, De Amescua Seco A, García Sánchez L, Pérez Cota M. Experiences in the application of software process improvement in SMEs. *Software Quality Journal* 2002; **10**(3):261–273.
3. Deming WE. *Out of the Crisis*, vol. xiii. Massachusetts Institute of Technology Center for Advanced Engineering Study: Cambridge, MA, 1986; 507.
4. Basili VR. *Quantitative Evaluation of Software Methodology*. Department of Computer Science, University of Maryland, 1985.
5. Paulk MC, Curtis B, Chrissis MB, Weber CV. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley: Reading, MA, 1995.
6. CMMI Product Developement Team. Capability Maturity Model Integration (CMMI) for systems engineering, software engineering, integrated product and process development, and supplier sourcing version 1.1. *CMMI-SE/SW/IPPD/SS, V1.1*, CMMI Product Development Team, 2002.
7. ISO/IEC. ISO/IEC TR 15504:1998—Software process assessment, 1998. http://www.sei.cmu.edu/iso-15504 [1 July 2004].
8. McFeeley B. IDEALSM: A user's guide for software process improvement. *Technical Report CMU/SEI-92-TR004*, Software Engineering Institute, Pittsburgh, PA, 1991.
9. Wiegers KE, Sturzenberger DC. A modular software process mini-assessment method. *IEEE Software* 2000; **17**(1):62–70.
10. Zahran S. *Software Process Improvement: Practical Guidelines for Business Success* (*SEI Series in Software Engineering*, vol. xxviii). Addison-Wesley: Reading, MA, 1998.
11. Kuilboer JP, Ashrafi N. Software process and product improvement: An empirical assessment. *Information and Software Technology* 2000; **42**(1):27–34.
12. Reifer DJ. The CMMI: It's formidable. *Journal of Systems and Software* 2000; **50**(2):97–98.
13. Scott L, Jeffery R, Carvalho L, D'Ambra J, Rutherford P. Practical software process improvement—the IMPACT project. *Proceedings of the 2001 Software Engineering Conference*, Australia, 2001. IEEE Computer Society Press: Los Alamitos, CA, 2001; 182–189.
14. Kautz K, Hansen HW, Thaysen K. Applying and adjusting a software process improvement model in practice: The use of the IDEAL model in a small software enterprise. *Proceedings of the 2000 International Conference on Software Engineering*. IEEE Computer Society Press: Los Alamitos, CA, 2000; 626–633.
15. El-Emam K, Goldenson D, McCurley J, Herbsleb J. Modelling the likelihood of software process improvement: An exploratory study. *Empirical Software Engineering* 2001; **6**(3):207–229.
16. Rainer A, Hall T. A quantitative and qualitative analysis of factors affecting software processes. *Journal of Systems and Software* 2003; **66**(1):7–21.
17. Herbsleb J, Zubrow D, Goldenson D, Hayes W, Paulk M. Software quality and the capability maturity model. *Communications of the ACM* 1997; **40**(6):30–40.
18. Herbsleb JD, Goldenson DR. A systematic survey of CMM experience and results. *Proceedings of the 18th International Conference on Software Engineering*. IEEE Computer Society Press: Los Alamitos, CA, 1996; 323–330.
19. Conradi R, Fuggetta A. Improving software process improvement. *IEEE Software* 2002; **19**(4):92–100.
20. Gorschek T, Wohlin C. Identification of improvement issues using a lightweight triangulation approach. *European Software Process Improvement Conference*, Graz, Austria, 2003.
21. Wohlin C, Runeson P, Höst M, Ohlson MC, Regnell B, Wesslén A. *Experimentation in Software Engineering: An Introduction* (*Kluwer International Series on Software Engineering*, vol. 6). Kluwer Academic: Boston, MA, 2000.
22. Robson C. *Real World Research: A Resource for Social Scientists and Practitioner-researchers* (2nd edn). Blackwell: Oxford, 2002.
23. Saaty TL. *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. McGraw-Hill: London, 1980.
24. Chu P, Liu JK-H. Note on consistency ratio. *Mathematical and Computer Modelling* 2002; **35**(9–10):1077–1080.
25. Apostolou B, Hassell JM. Note on consistency ratio: A reply. *Mathematical and Computer Modelling* 2002; **35**(9–10):1081–1083.
26. Saaty TL, Vargas LG. *Models, Methods, Concepts & Applications of the Analytic Hierarchy Process* (*International Series on Operations Research & Management Science*, vol. 34). Kluwer Academic: Boston, MA, 2001.
27. Beck K, Fowler M. *Planning Extreme Programming* (*The XP Series*). Addison-Wesley: Boston, MA, 2001.
28. Leffingwell D, Widrig D. *Managing Software Requirements: A Unified Approach* (*Object Technology Series*). Addison-Wesley: Reading, MA, 2000.
29. Karlsson J, Wohlin C, Regnell B. An evaluation of methods for prioritizing software requirements. *Information and Software Technology* 1998; **39**(14–15):939–947.