An Industrial Survey of Requirements Interdependencies in Software Product Release Planning

Pär Carlshamre[a], Kristian Sandahl[a], Mikael Lindvall[b], Björn Regnell[c], Johan Natt och Dag[c]
[a]Ericsson Radio Systems AB, Sweden, {par.carlshamre, kristian.sandahl}@era.ericsson.se
[b]Fraunhofer USA Center for Experimental Software Engineering, mlindvall@fraunhofer.org
[c]Dept. of Communication Systems, Lund University, Sweden, {bjorn.regnell, johan.nattochdag}@telecom.lth.se

Abstract

The task of finding an optimal selection of requirements for the next release of a software system is difficult as requirements may depend on each other in complex ways. This paper presents the results from an in-depth study of the interdependencies within 5 distinct sets of requirements, each including 20 high-priority requirements of 5 distinct products from 5 different companies. The results show that (1) roughly 20% of the requirements are responsible for 75% of the interdependencies, (2) only a few requirements are singular, (3) customer-specific bespoke development tend to include more functionality-related dependencies whereas market-driven product development have an emphasis on value-related dependencies. Several strategies for reducing the effort needed for identifying and managing interdependecies are outlined. A technique for visualization of interdependecies with the aim of supporting release planning is also discussed. The complexity of requirements interdependency analysis is studied in relation to metrics of requirements coupling. Finally, a number of issues for further research are identified.

1. Introduction

As incremental systems development strategies become commonplace in industry, increment planning (or release planning, the words are used interchangeably herein) gains both importance and interest. The task of scheduling an optimal selection of requirements for a particular increment is as complex as it is important [2, 6, 8]. An informal survey involving 8 managers responsible for increment planning at Ericsson Radio Systems revealed six different planning parameters that has to be considered and satisfied: *available resources, delivery time, requirements interdependencies, requirements priority, system architecture, and dependencies to the code base.*

Priority of requirements is a major determinant in increment planning, but the fact that requirements are related to each other makes it difficult, or sometimes even impossible, to schedule requirements based on priority only. Not much is known about the nature and frequency of requirements interdependencies, and further research is requested [6]. Even though there are examples of research tools that can manage various types of *requirements interactions* [4], (see [9] for a comprehensive overview of Requirements Interaction Management), none of them seem to focus on such interactions from a release planning perspective. Nor do popular commercial RE tools offer such capabilities.

To gain knowledge about requirements interdependencies, a survey of five different companies was made. In each case, a requirements manager responsible for a particular product or project was asked to identify and classify interdependencies among 20 high priority requirements in his repository. Our aim was to learn about the nature of interdependencies in general, to be able to classify them and asses the relative frequency of different classes, in order to be able to support the task of release planning.

To the same end, a technique for simple visualization of the requirements interdependencies was then applied to each of the five cases. Our preliminary studies indicate that visualization of requirement interdependencies is efficient in supporting the identification of salient characteristics of a set of requirements, from a release planning perspective.

In this paper we first outline the five cases and the operational details of the survey. We then present the results and offer an interpretation of the findings. This is followed by a description of the visualization technique together with a discussion of what can be gained from visualizing interdependencies. Finally we end with some concluding remarks and requests for further research.

2. Survey planning and operation

Five different companies within different market segments were involved in the survey. At each company, a senior project manager, product manager or requirements engineer was engaged in the study. For simplicity, these are referred to as "requirements managers", or RM.

The five cases. Three of the cases can be labelled as product development, meaning that there was a mature product out on the market, and the development situation can be described as incremental and market-driven. The remaining

two cases can be labelled as bespoke, meaning that there was no released version of the product on the market, and there was a specific customer paying for the development. The following summarizes relevant facts about the cases:

- Case 1: A small company with one product and services associated with it. The product is a support tool for requirements management with special attention on prioritization. It is a mature product which has been available for several years. As could be expected, the staff is highly experienced in requirements management and they have an explicit focus on RE processes.
- Case 2: A medium sized company with three different product areas. The product involved in our survey is a system for storing and management of X-ray images for medical services. It is mature and has been on the market for several years. The RM is experienced in requirements management issues, and the unit has an explicit and mature RE process.
- Case 3: A product unit within a large telecommunications company. The product is a tool for analysis of large data tables in telephone exchanges, and has been available on the market for three years. The company in general has a high level of process awareness, and the requirements management team has several years of experience. They have an explicit and mature RE process.
- Case 4: A software engineering research institute. The project has a clear commercial aim aside from the research focus, and will result in a decision support system handling large quantities of company and market information. The RM is highly experienced within requirements management, but the focus on the RE process is implicit in this project.
- Case 5: A medium sized company with a product department and a services department. The product is a document management platform, and the project in our survey (belonging to the services department) is aimed at adapting and extending the platform for a specific customer. The RM is a highly experienced project manager; the focus on RE processes is implicit.

Data collection. In each case, the RM was asked to randomly select 20 high priority requirements from the current requirements repository or requirements specification. Priority was the only selection criteria, and they were explicitly instructed not to consider potential dependencies during selection. The requirements were entered into a specially designed spreadsheet, allowing pairwise consideration of all 20 requirements (see Figure 1).

Each session took about 4 hours, and started with a description of the aim of the study, and a discussion of the general development situation. Then, the search for dependencies started with a consideration of requirement 1 and 2, 1 and 3, 1 and 4 and so forth. For each pair of requirements there were a number of considerations to make, resulting in one of the following cases:

1. No relation whatsoever could be found. In this case the RM was asked how certain he was of this, on the scale {Possibly, Probably, Positively}
2. A relation was found. In this case the respondent could choose one of five types of interdependencies (explained below), or add any new type if needed. He was also asked to rate the certainty, similar to the previous case.

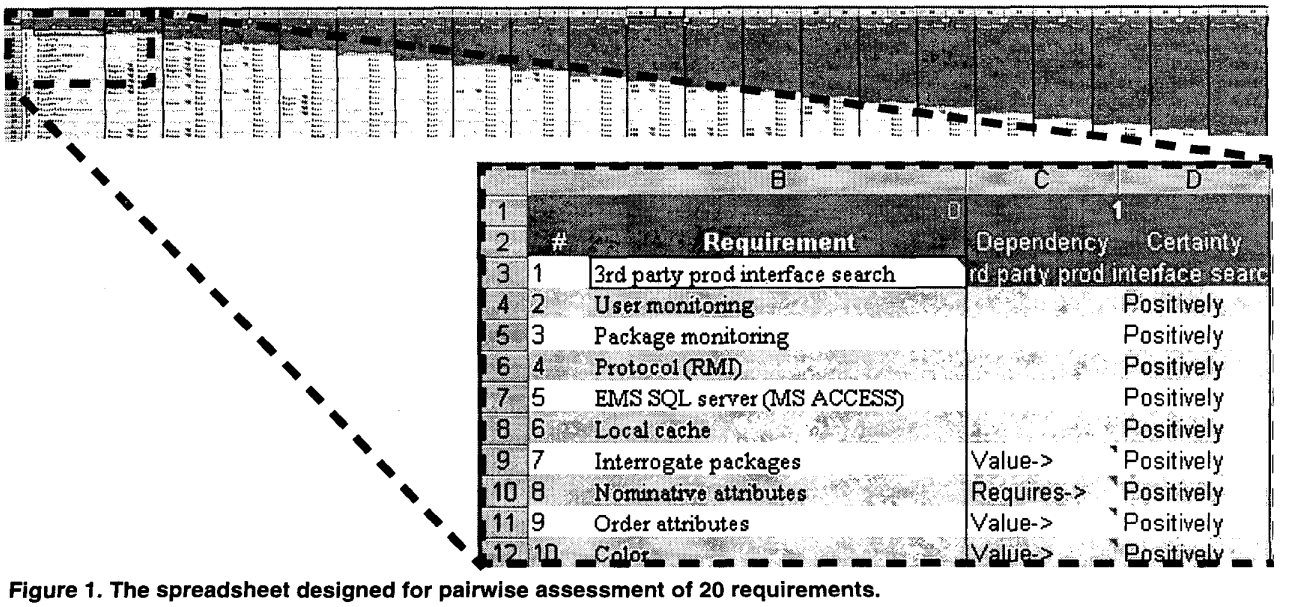


Figure 1. The spreadsheet designed for pairwise assessment of 20 requirements.

For each interdependency that was found, a comment of a more qualitative nature was inserted in the spreadsheet, in order to allow further analysis.

2.1. Types of interdependencies

Based on previous interviews with requirements engineers at two different companies, and on a proposal given in [6], a preliminary set of interdependency types were created, as shown in Table 1. Since this set was only preliminary, the RM could add any type of relationship that was discovered. The following examples may help to clarify the practical meaning of the five types.

Example 1: AND. A printer requires a driver to function, and the driver requires a printer to function.

Example 2: REQUIRES. Sending an e-mail requires a network connection, but not the opposite.

Example 3: TEMPORAL. The function *Add object* should be implemented before *Delete object*. (This type is doubtful, which is discussed in section 3.1)

Example 4: CVALUE. A detailed on-line manual may decrease the customer value of a printed manual.

Example 5: ICOST. A requirement stating that "no response time should be longer than 1 second" will typically increase the cost of implementing many other requirements.

Example 6: OR. In a word processor, the capability to create pictures in a document can either be provided as an integrated drawing module or by means of a link to an external drawing application.

In some cases, more than one relationship could be identified between two particular requirements. For instance, it is intuitive that if a requirement R_1 requires another, R_2, to function, R_2 will also increase the value of R_1 (from zero). To solve this, the interdependencies were given a priority, according to Table 1, and only the interdependency with the highest priority was recorded. The priorities were derived from the following reasoning:

- In general, functional interdependencies should have higher priority than value-related.
- Since an AND relationship consists of a REQUIRES relation in both directions, AND has priority over REQUIRES
- TEMPORAL was considered to be more "functional" than "value-related", often expressing a situation where it is impossible or very difficult to correctly implement and test R_2 until R_1 is done. Therefore it was given priority over CVALUE and ICOST.
- CVALUE and ICOST should have the same priority, and in case of a conflict, they have to be traded-off against each other.

OR seemed to be the least important type, since implementing both R_1 and R_2 would affect neither function nor customer value. One could argue that since this would affect the overall cost for the development organization, OR should have the same priority as ICOST. However, we concluded that even though the *identification* of an OR interdependency may be important, it will probably result in the elimination of one of the requirements (implying XOR rather than OR), which in turn will solve the problem from a release planning perspective.

3. Results and analysis

3.1. Qualitative results

At the beginning of each session, the various types of interdependencies were explained and discussed. The general opinion among the RMs was that the types were relevant and intelligible, and they could easily find examples from their own requirement domains. Although the RMs had the option of adding new types of interdependencies at any stage, no new types were discovered in our survey. However, at several points questions about the current set of interdependencies arose, concerning definitions and borderline cases. For example, "*if R_2 is completely worthless to the customer without R_1, and we would thus never do R_2 without R_1, do we classify the relationship as REQUIRED or just CVALUE?*" In general, we let the RMs decide to a great extent what was reasonable and practical. In this particular case the decision was to define AND and REQUIRES relationships as strictly functional, i.e., in the above example we would have a CVALUE interdependency after all.

At the end of each session there was a brief discussion about the origin of the interdependencies, and some of the insights are summarized below.

Table 2. Preliminary set of interdependencies.

Priority	Type	Meaning
1	R_1 AND R_2	R_1 requires R_2 to function, and R_2 requires R_1 to function.
2	R_1 REQUIRES R_2	R_1 requires R_2 to function, but not vice versa.
3	R_1 TEMPORAL R_2	Either R_1 has to be implemented before R_2 or vice versa.
4	R_1 CVALUE R_2	R_1 affects the value of R_2 for a customer. Value can be either positive or negative.
4	R_1 ICOST R_2	R_1 affects the cost of implementing R_2. Value can be either positive or negative.
5	R_1 OR R_2	Only one of $\{R_1, R_2\}$ needs to be implemented.

AND usually emanates from a parent-child relationship where R_1 and R_2 are subordinate to a higher-level requirement R. This is a common break-down of a requirement, and as a consequence, AND type interdependencies can usually be traced in a requirements management tool. However, this was not done in any of the cases.

REQUIRES sometimes arises from the opposite reasoning: "If we do R_2, then we can do R_1 too!", which implies that the direction of the relationship should be the opposite. Also from a visualization perspective (which will be discussed later), it may be more natural to draw attention to the requirement that enables, or *is a prerequisite for* another one, than the opposite. With this logic, the visual semantics are also more similar to the other types of dependencies, e.g., R_1 affects R_2. We have tried both logics and prefer to visualize the IS-PREREQUISITE-FOR logic, which is also used in Figure 3. No controlled user evaluations have been conducted so far, though.

TEMPORAL dependencies seemed relevant at the outset, but we soon learned that they are rarely interesting, since they usually can be seen as either a REQUIRED dependency, or an ICOST dependency. Thus, no definite temporal interdependency was found in any of the five cases. If, for example, *Add object* really needs to be implemented before *Delete object*, it is evident that *Delete object* REQUIRES *Add object*. In some cases, temporal interdependencies express process knowledge rather than product knowledge, i.e., "It's smarter to do this before we do that". In these cases the relationship is a clear ICOST type interdependency. In either case, these interdependencies are usually handled by including both requirements in the same increment.

CVALUE dependencies are important from a product planning perspective. For example, the product management may want to level out the value added over several releases, in order to maintain a continuous interest from the customers, rather than putting all the best features in one release. CVALUE and ICOST dependencies are sometimes found in combination. In some cases R_1 increases the value of R_2 for the customer, but it also increases the cost of implementing R_2. In practice, it seems as if only two types are interesting: either R_1 increases the value of R_2 for the customer *more* than it increases the cost of R_2 for the developer, or vice versa. Obviously, all assessments of value-related interdependencies are subjective, and it is sometimes difficult to state whether the extra cost of implementation exceeds the extra value for the customer or vice versa. On the other hand, these types of decision are made, and have to be made, by product committees regardless of how difficult they are.

Since the OR relationship means that "either we could do R_1 or we could do R_2", an OR dependency usually indicates that further investigations are needed. At some occasions, what initially seemed to be an OR interdependency turned out to be irrelevant for requirements management; From a users point of view it can mean two different, but equally important, ways of doing the same thing, and thus the two requirements should be treated as separate.

We had expected to find *conflicting* requirements in our survey, since this type of dependency is reported to be common [3, 9]. However, no such interdependencies were identified, and the reason for this may be that conflicts had been eliminated at this stage in the requirements management process of the five cases.

As described previously, we also collected data about how certain the RMs were in each pairwise assessment. The idea behind this was to see whether a future support tool would need to manage fuzziness even along this dimension, i.e., "I think there is a dependency here, but I'm not sure". However, the degree of certainty throughout the five cases was so overwhelming that it turned out to be of little interest. In no case the ordinal value of average certainty was below 2.8 out of 3.

From the debriefing sessions, it was also clear that the RMs considered the excercise valuable for their work; not only did they find the particular dependencies, but they also discovered other problems with their requirements. In one case, for example, it was discovered that a requirement was missing in order to be able to implement two other. The general opinion could be summarized by the following quote: *"It's always important to look at your requirements from different perspectives. This is a new perspective and it provides a structured approach to scrutinizing your requirements."*

3.2. Quantitative results

Theoretically, there can be 190 pair-wise interdependencies among 20 requirements. In our survey, the number of identified interdependencies varied between 19 and 42. In

Table 2. Summary of identified interdependencies.

	# dependencies	most common type	# singular req's	10% of the req's are responsible for	20% of the req's are responsible for	coupling (cf. section 3.5)
Case 1	19	ICOST 79%	4	47% of dep's	79% of dep's	10%
Case 2	29	CVALUE 45%	3	55%	76%	15%
Case 3	42	ICOST 86%	3	50%	74%	22%
Case 4	41	AND 41%	3	44%	71%	22%
Case 5	24	REQUIRES 79%	4	42%	67%	13%

Case 1, 2 and 3, the most common type of interdependency was value-related, i.e., either ICOST or CVALUE. In these cases such interdependencies were responsible for between 59% and 90% of all identified relationships. In Case 4 and 5, the most common type was functionality-related, i.e., either AND or REQUIRES (see Table 2). It is interesting to note the difference between product-oriented cases and bespoke development. In the product development cases, the value-related interdependencies are more common than functionality-related, and vice versa. A χ^2-test reveals that the difference is significant on the 0.05% level (see Table 3).This corresponds well with the intuition that functionality, or getting things to work, is more crucial in the first increment (which a bespoke situation can be said to resemble), and adding value or reducing cost is more an issue in later increments

As can also be seen from Table 2, there is only a few requirements that are completely singular, i.e., has no relationship with any other requirement. Furthermore, a few requirements are responsible for a large part of all interdependencies. These findings are important .and will be returned to later, in connection with the implications for increment planning.

Table 3. Value-related interdependencies dominate in product development, and functionality-related ditto in bespoke development.

	Product dev. Case 1-3	**Bespoke dev.** Case 4-5
Value-related	75	21
Func. related	15	44
Σ	90	65

3.3. Visualization of interdependencies

As mentioned in the introduction, requirements interdependencies has an important role to play in release planning. But interdependencies are rarely identified explicitly. The sheer number of them makes them difficult to identify and manage, and the complexity growth is exponential. Furthermore, interdependencies are fuzzy, in the sense that a relationship can be more or less important to consider. Even if R_1 affects the value of R_2, for example, the influence can be large or insignificant. Thus, interdependencies can only be treated formally and automatically to a certain extent for practical purposes. The important decisions are left to the requirements engineers. Considering the complexity of the matter, the fuzziness of the interdependencies, and the focus on supporting human decisions has inspired us to explore a data visualization approach to managing requirements interdependencies.

By representing requirements and their interdependencies by objects and arrows in a traditional form it is possible to draw important conclusions associated with release planning from just a glance at the graph (see Figure 2). First of all, singular requirements, i. e., requirements having no relationship with any other requirements, are easily spotted. These can be scheduled for any increment, from an interdependencies point of view. They can be used to "top off" an increment so that it fits the amount of available development resources, for example.

Second, requirements having relationships to many other requirements are also easily identified. These should be scheduled for early increments, in order to reduce risk.

Finally, free clusters of requirements can be scheduled for any increment, as long as all involved requirements are scheduled for the same increment. In Figure 2, requirements 13, 15 and 16 represent a cluster from a functional standpoint. Their respective customer value is affected by requirement 4.

We have only begun to explore the possibilities of using visualization of interdependencies to support release planning, but the examples from our survey encourage us to further investigate the power of this approach.

3.4. Supporting identification of interdependencies.

For obvious reasons, industrial software developers are reluctant to adding new analysis activities to an already strained project timetable. Therefore it is necessary to address the issue of the efforts needed to manage interdependencies, and how these efforts could be reduced.

In our study, it took between 2.5 and 3 hours to do the pairwise assessment of 20 requirements (190 assessments), which means that it took approximately 1 minute per assesment. For 20 requirements this may be considered as reasonable, but the required effort increases dramatically with the number of requirements. With 40 requirements, for example, it would take in the vicinity of 12 hours. Thus it is important to find ways of reducing the number of assessments, and our preliminary investigations indicate a number of approaches with varying efficiency.

Identifying singular requirements. In our survey, each of the cases had roughly 20% singular requirements. If these could be identified at an early stage, the number of assessments needed would be reduced substantially from

$$\sum_{1}^{(n-1)} i = \frac{n(n-1)}{2}$$

to

$$\sum_{1}^{(n-1)-s} i = \frac{(n-s)(n-1-s)}{2}$$

(where n is the number of requirements considered, and s is the number of singular requirements) and the gain would be

$$\frac{s(2n-1-s)}{2}$$

For example, by identifying 4 singular requirements out of 20, we can eliminate 70 assessments of the 190, which is a 37% decrease. Of course, identification of singular requirements needs some effort too, but in our study these were quite easily spotted, and with a high degree of certainty, by the RMs. Furthermore, the consideration of whether a particular requirement is singular or not is less time-consuming than n-1 pairwise assessments.

Scanning for similarity. An issue related to interdependencies is that of similarity between requirements. If two requirements have similar slogans, one can suspect that there is a functional relationship between them. The connection between similarity and interdependency is evident in the case where we have two requirements R_1 and R_2, with the exact same formulation. In this case we would have an OR interdependency between R_1 and R_2. But even in other cases there are commonalities. If, for example, R_1 stated that "It shall be possible to sort by name and address" and R_2 stated that "It shall be possible to sort by age", it is likely that it would be wise to treat these requirements at the same time, to save development resources. This would be an ICOST interdependency, according to the previous classification scheme.

In order to investigate whether a similarity analysis could support the identification of interdependencies in a requirements repository, a lexical analyzer was applied to the five different sets of 20 requirements. The 100 requirement slogans was relieved of words like "and", "of", "shall" etc., and then fed into the lexical analyzer (see [5] for a complete description).

A χ^2-test of the covariance of lexical similarity and actual dependencies gave a p-value of <0.0001, which shows that the similarity measure used vary significantly with actual dependencies. Although the accuracy and reliability is not sufficient for lexical analysis to be used as an exclusive identification technique, its low cost and high degree of automation may justify its use in conjunction with other techniques. Again, the reader is referred to [5] for details.

Identifying highly dependent requirements. A powerful approach to reducing the number of pairwise assessments is to first identify highly dependent requirements, and then perform the pairwise assessment with only these. Our data suggests that by identifying the 20% "most dependent" requirements, one can cover between 67% and 79% of all dependencies (see Table 2). As a consequence, with d highly dependent requirements we need only make
pairwise assessments (cf. [10], where so called Root Requirements are identified in order to avoid the N^2 problem). Hence, in our study where 20% equals to 4 require-

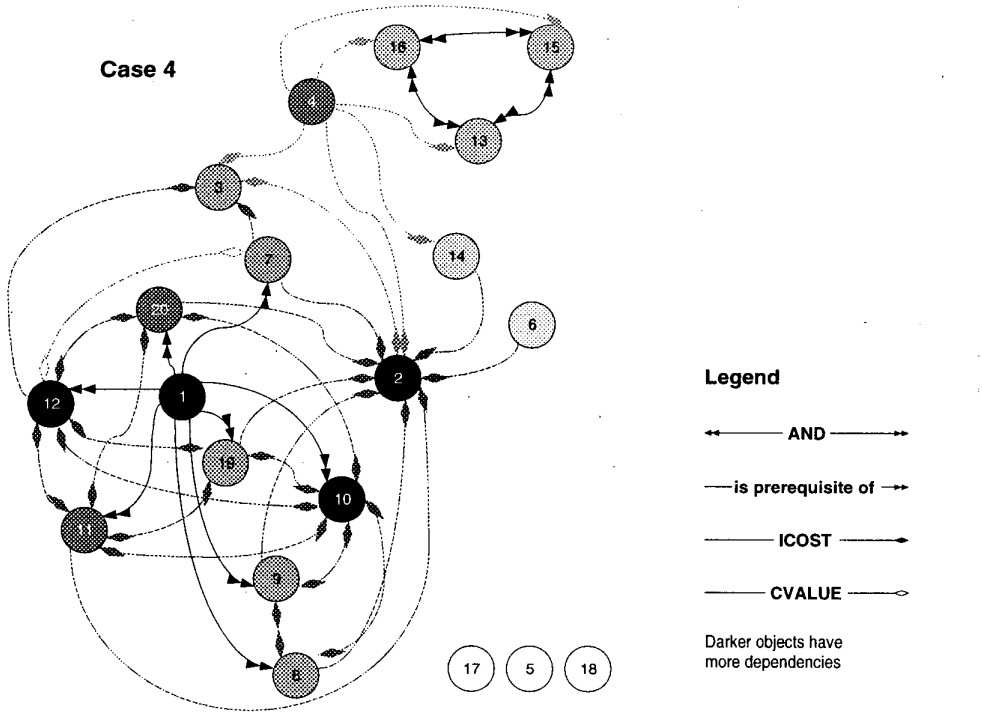


Figure 2. Visualization of requirements interdependecies for one of the five cases. Positive and negative value impacts are illustrated by means of color in the original illustration.

$$\sum_{n-d}^{n-1} i = \frac{d(2n-1-d)}{2}$$

ments, 70 pairwise assessments (demanding slightly over one hour) would have covered roughly 3/4 of the interdependencies. This may be a reasonable trade-off in many industrial settings.

Our material does not indicate how easy or difficult it is for an RM to identify highly dependent requirements, and this will be one of the subjects of future studies. However, most of the highly dependent requirements in our survey fall in one of the following categories:

- Migration to a new platform or OS.
- Changes to core functionality.
- Changes to core data structures.
- Major changes to user interface.

It is consistent with intuition that these types of requirements would affect many other requirements, as well as the code base (unless the design was made with such changes in mind, explicitly). In our experience requirements engineers are fairly capable of pointing out requirements that will have a large impact of other parts of a system, as part of the cost estimation procedures [7]. Thus we believe that identification of the 20% most dependent requirements could be a surmountable problem, especially if supported by heuristics of the kind listed above.

3.5. An interdependency measure

Selecting a number of requirements for realization in a particular release can be viewed as partitioning the complete set of requirements in an optimal way. From an interdependencies point of view, the strategy should be to split the set of requirements in such a way that there are few interdependencies between the partitions, thus making the releases as independent of each other as possible. Similarly, requirements that have many interdependencies between them should be scheduled for the same release in order to simplify realization. This is important not only for functional, but also for cost-driving interdependencies (ICOST), in order to minimize risk.

This situation is similar to the software design strategies known as *low coupling* [1, 11], and it is tempting to borrow this concept from the domain of structured and object-oriented design. Coupling is a measure of the strength of association between entities (e.g., objects). Basically, low coupling between entities promotes a high degree of modularity and encapsulation, which is desirable.

The same reasoning and terminology could be applied to the problem of partitioning a set of interdependent requirements. The number of possible distinct interdependencies in a set of R requirements is $R(R\text{-}1)/2$. If we let I be the number of distinct interdependencies within a specific set of R requirements, we may define the degree of **requirements coupling** as:

$$Creq = \frac{I}{(R(R-1))/2}$$

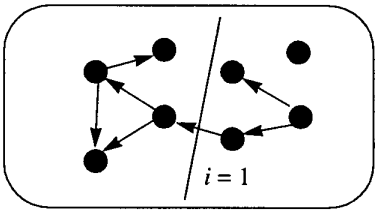
Requirements coupling could be seen as an indicator of the complexity of the planning problem. In our survey, the requirements coupling factor $Creq$ varies between 10% and 22% (cf. Table 2).

Furthermore, if we let i be the number of interdependencies between two partitions (see Figure 3), we may define the degree of **release coupling** as

$$Crel = \frac{i}{I}$$

Thus, from an interdependencies perspective, a strategy for release planning would be to find a partitioning that minimizes release coupling.

Figure 3. Example illustrating the concepts of requirements and release coupling.



In object-oriented design, the concept of coupling between entities is an important quality aspect, as an indicator of several important characteristics. According to [11], a large number of couples indicates a higher sensitivity to changes in other parts of the design and is also detrimental to the comprehensibility of a module. It is reasonable to believe that this would apply to requirements and release planning too. Changes to requirements that are highly dependent are likely to affect many other requirements, and a set of singular requirements are certainly easier to manage and comprehend than a set of highly dependent requirements. Thus, we believe that the concept of requirements coupling and release coupling is a fruitful one, and we encourage further exploration of its descriptive powers.

4. Conclusions and future research

Release planning is a crucial activity in market-driven software development, because it decides what should be delivered when. In an ideal situation, where there are no interdependencies between requirements, release planning is a matter of prioritizing the requirements, and then selecting a number of top priority requirements, depending on the available resources and the delivery date at hand. However, in our survey of five independent organizations' requirements repositories, we found that only about 20% of the require-

ments are singular, which is far from the ideal situation. This emphasizes the need for exploring requirements interdependencies, and also a need to support identification and management of interdependencies.

In our survey, we also found that relatively few requirements are responsible for a large share of the interdependencies. In fact, 20% of the requirements were involved in roughly 75% of all interdependencies. Thus, by supporting the identification of these highly dependent requirements, it is possible to reduce the total effort needed to identify and manage interdependencies in a requirements repository.

We proposed a classification scheme for interdependencies, including functional (AND, REQUIRES) and value-related (ICOST, CVALUE), and found that there tend to be more functionality-related interdependencies in a bespoke development situation, whereas there are more value-related interdependencies in a product development situation. This corresponds well with the intuition there is more focus on getting things to work than adding value in the bespoke situation, and vice versa.

Aiming at supporting release planning, a simple visualization technique was then applied to the requirements and their interdependencies in our survey. This technique proved to be very powerful to support reasoning about possible and good ways of partitioning a set of requirements. The visualization allowed for identification of singular requirements, clusters of interdependent requirements, as well as highly dependent requirements at a quick glance. This has implications for future support tools for release planning.

Finally, we compared the issue of interdependencies among requirements as well as between partitions of a set of requirements to the concept of coupling in structured and object-oriented design. We offered definitions of requirements coupling and release coupling, which may serve as interdependency goodness measures in release planning.

The findings from our survey give rise to several questions, and indicate needs for future research in many directions: First of all, it is important to relate our results, as well as our classification scheme, to studies of other domains and sample sizes. For this purpose we have made available a spreadsheet for pairwise assessment of the kind described herein, together with instructions[1]. Second, to minimize the effort needed to identify and manage requirements interdependencies, it is important to find and refine heuristics for identification of singular as well as highly dependent requirements.Third, we have only begun to explore the possibilities of visualizing requirements interdependencies, and we encourage further studies in this area, to find out what is possible and what is practical. Fourth, the concepts of requirements coupling and release coupling and their usefulness and applicability needs further treatment. Finally, we have not related the issue of requirements interdependencies to that of dependencies between requirements and the code base, i.e., already implemented requirements. In order to support release planning, these dependencies may be equally important, and further studies of the subject are needed.

1. Available at http://www.ida.liu.se/~parca/interdependencies.html

Acknowledgements

The authors would like to thank the anonymous requirements managers that participated in the survey. We would also like to thank Dr. Joachim Karlsson and the anonymous reviewers for their valuable contributions to this paper.

References

[1] Briand, L.C., Daly, J.W., Wüst, J.K. A Unified Framework for Coupling Measurement in Object-Oriented Systems, *IEEE Trans. on Software Engineering*, 25(1), June, 1994.

[2] Carlshamre, P., Regnell, B. Requirements Lifecycle Management and Release Planning in Market-driven Requirements Engineering Processes. *IEEE Int. workshop on the requirements engineering process. In Proc. 11th Int. Conf. on Database and Expert Systems Application DEXA2000.* September 2000, Greenwich, UK.

[3] Egyed, A., Boehm, B., A Comparison Study in Software Requirements Negotiation, *Proc. of the 8th Annual International Symposium on Systems Engineering* (INCOSE'98), 1998.

[4] Feather, M.S., Cornford, S.L., Gibbel, M. Scalable Mechanisms for Requirements Interaction Management. *Proc. Int. Conference on Requierements Engineering*, IEEE, 2000.

[5] Natt och Dag, J., Regnell, B., Carlshamre, P., Andersson, M., Karlsson, J., Evaluating Automated Support for Requirements Similarity Analysis in Market-Driven Development, *7th Int. Workshop on Requirements Engineering.* (REFSQ'2001), Interlaken, Switzerland, June 2001.

[6] Karlsson, J., Olsson, S., Ryan, K. Improved Practical Support for Large-scale Requirements Prioritising. *Requirements Engineering Journal*, 2(1):51-60, 1997.

[7] Lindvall, M. and Sandahl, K. How Well do Experienced Software Developers Predict Software Change? *Journal of Systems and Software*, 43(1):19 - 27, 1998.

[8] Regnell, B., Beremark, P., Eklundh, O., A Market-driven Requirements Engineering Process - Results from an Industrial Process Improvement Programme. *Requirements Engineering Journal*, 3(2):121-129, Springer, 1998.

[9] Robinson, W.N., Pawlowski, S.D., Volkov, V. Requirements Interaction Management. GSU CIS Working Paper 99-7. Downloaded April 9 2001 from http://cis.gsu.edu/~wrobinso

[10] Robinson, W.N., Pawlowski, S.D. Surfacing Root Requirements from Inquiry Cycle Requirements. *Proc. Int. Conference on Requirements Engineering,* IEEE, April 6-8, 1998.

[11] Rosenberg, L.H., Hyatt, L.E. Software Quality Metrics for Object-oriented Environments. *Crosstalk Journal*, 10(4): April 1997, Software Technology Support Center, Hill, UT.