Petri Kettunen. 2006. Troubleshooting large-scale new product development embedded software projects. In: Jürgen Münch and Matias Vierimaa (editors). Proceedings of the 7th International Conference on Product-Focused Software Process Improvement (PROFES 2006). Amsterdam, The Netherlands. 12-14 June 2006. Springer. Lecture Notes in Computer Science, volume 4034, pages 61-78.

Troubleshooting Large-Scale New Product Development Embedded Software Projects

Petri Kettunen

Nokia Corporation
P.O. Box 301, 00045 NOKIA GROUP, Finland
petri.kettunen@nokia.com

Abstract. Many modern new product development (NPD) embedded software projects are required to be run under turbulent conditions. Both the business and the technological environments are often volatile. Uncertainty is then an inherent part of the project management. In such cases, traditional detailed up-front planning with supporting risk management is often inadequate, and more adaptive project management tools are needed. This industrial paper investigates the typical problem space of those embedded software projects. Based on a literature survey coupled with our practical experiences, we compose an extensive structured matrix of different potential project problem factors, and propose a method for assessing the project's problem profile with the matrix. The project manager can then utilize that information for problem-conscious project management. Some industrial case examples of telecommunications products embedded software development are illustrated.

1 Introduction

Most new electronic products contain embedded software in particular to enable more intelligent features and flexibility [1]. Thus, there will be more and more software projects developing embedded software for such new product development (NPD) markets.

Managing those modern industrial NPD projects successfully requires situation-aware control with the possible and oncoming troubles, taking the anticipated and even unexpected situational conditions into account [2]. Uncertainty is inherent [3, 4]. Project risk management is a traditional way of handling the obstacles, which may affect the project success adversely [5-7].

In this paper our premise is that in turbulent industrial business environments the product development projects must typically work under imperfect conditions. For example, it is hardly ever possible to avoid all external schedule pressures. In other words, the project management faces some problems all the time, and the project may be in some trouble even from the very beginning. This is sometimes referred to as project issue management [8]. In practice both proactive risk management as well as reactive problem (issue) management are needed [9].

The first step of problem-aware project management is to be able to recognize the current project problem factors. Project problems and uncertainties should be actively searched [10, 11]. There are no standard solutions, since the actual unique project context has to be taken into account.

J. Münch and M. Vierimaa (Eds.): PROFES 2006, LNCS 4034, pp. 61 – 78, 2006.

The purpose of this paper is to propose focused aids for identifying and evaluating the typical problem factors of large-scale NPD embedded software projects (such as telecommunications equipment). The rest of the paper is organized as follows. Chapter 2 explores the background and related work, and sets the exact research questions. Chapter 3 then describes our solution ideas, while Chapter 4 evaluates them. Finally, Chapter 5 makes some concluding remarks, and outlines further research ideas.

2 NPD Embedded Software Project Problems

2.1 Typical Software Project Problem Factors

Over the years, there have been numerous investigations about typical software project problems and failure factors. Table 1 lists some of the known ones (ordered by the year of publication). For more, see for example [6, 8, 12-19].

Table 1. A survey of software project problems, risks, and failure factors

Investigation	Distillation
Brooks [20]	Fundamental problems of software engineering management
Curtis, et al. [21]	Human and organizational factors affecting productivity and quality of large projects (including embedded systems)
Boehm [5]	Top 10 general software project risk items
McConnell [22]	36 “classic” software project mistakes; Common schedule risks
McConnell [23]	Software project “survival test”; Checklists
Royce [24]	Top 10 risks of “conventional” process
Brown [25]	Typical software project management malpractices and pitfalls
Ropponen, et al. [26]	Categories of software project risks and their influencing factors
Schmidt, et al. [27]	Systematic classification of empirically observed project risk factors
Smith [28]	40 root causes of software project failure
May, et al. [29]	Common characteristics of dysfunctional software projects
Fairley, et al. [30]	10 common software project problem areas and some antidotes

It is possible to categorize different project problem factors from various different points of view. For example the classic SEI taxonomy defines one way of categorizing common risk factors under project environment, product engineering, and program constraints [31]. Other alternatives are for example in [22, 26, 27, 32].

It is in addition important to understand that in complex (multi)project environments the project problems do not usually manifest themselves in isolation, but there are often multiple overlapping problems at the same time. Furthermore, there are often complex cause-effect relationships of the different problem factors, i.e., a single problem may have adverse additional consequences [32/Ch. 5, 33/Ch. 3].

2.2 Embedded Software Project Concerns

Compared to traditional software projects, embedded systems introduce certain additional intrinsic software development problems. There are both software engineering technical and management challenges [1, 21].

Figure 1 illustrates those many potential sources of problems. Notably many problems really stem from the software project external reasons and dependencies.

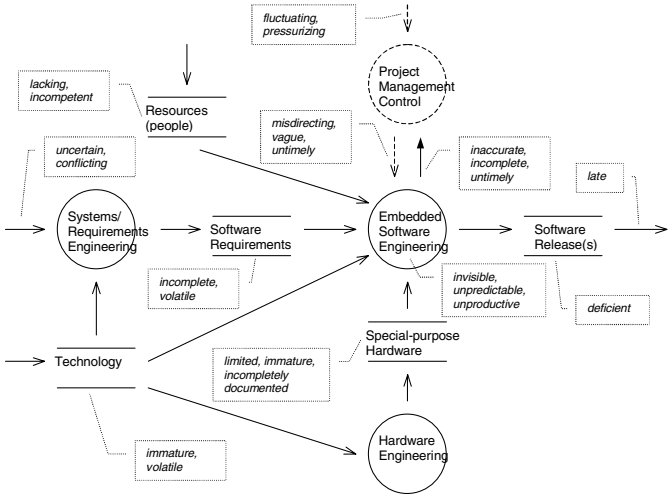

Fig. 1. Some embedded software project problem sources

Those special problem factors of embedded software projects have not been investigated especially widely in the literature. For some related studies, see for example [34-36]. Many embedded software project problems originate fundamentally from knowledge management issues [37].

2.3 NPD Software Project Characteristics

The development of new market-driven commercial products creates additional special characteristics of the software project environment. Figure 2 illustrates a typical NPD environment: The embedded software project team is an element of it. The NPD environment is not fundamentally different from other software development contexts. However, the emphasis on business drivers and product innovation management put considerable weight on certain problem areas in particular in large organizations.

The embedded software project teams working in such environments often face many sources of turbulence [4, 38]. The company, responding to emerging and fluctuating market needs, has to manage its product development portfolio (aggregate

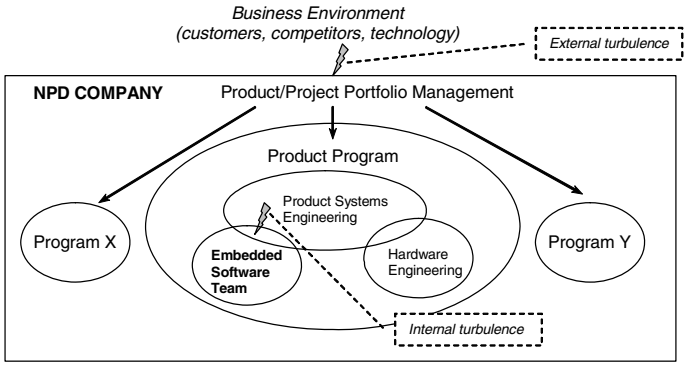

Fig. 2. Embedded software project team NPD context

project plan) accordingly [39/Ch. 2]. This may consecutively introduce various changes to the embedded software project teams (e.g., product features, releases schedules, project resource allocation). In addition, the other internal parts of the product development program (e.g., concurrent hardware engineering) may cause changes to the software part. It is important to understand the true nature of the project and the success criteria, and to incorporate the embedded software development as an integral part of the overall product system development [35, 40].

The problems of NPD projects have gained increasing research interests due to the current major transitions in many product development areas (e.g., telecommunications industry). A seminal survey of NPD literature is presented in [25]. An integrative model of different contributing product development project success factors is constructed. Ernst makes a critical summary of the NPD success factors empirical research results [41]. Notably there is no universal definition of "success". Recently for example Cooper, Edgett and Kleinschmidt survey the general success/failure factors [42]. In general, software new product development can be seen as a series of questions and problems to be solved [11].

2.4 Research Questions

Based on the background presented in Ch. 2.1-2.3, we now set the following specific research questions:

1. How to recognize the typical problems of large-scale NPD embedded software projects?
2. How to assess the feasibility and achievability ("health") of such projects?

Answering the former question brings insight to the latter one. By recognizing the particular alerting problem areas, the project manager can conduct and steer the project rationally, even under considerable trouble conditions.

The rest of this paper proposes pragmatic aids for answering those questions in a systematic way. The research method is constructive, based on the literature surveys

coupled with our own practical experiences with large-scale embedded software development for new telecommunications products. Our primary scope is in break-through development projects, creating entirely new market-driven products for the organization. Note that project financial issues (such as budgeting, rewarding) are excluded.

3 Troubleshooting NPD Embedded Software Projects

3.1 Project Problem Profiler

Our proposition for recognizing and evaluating the project problem issues is a matrix of typical problem factors and their likely impacts. Table 2 illustrates the overall structure of the matrix (see Appendix for the complete table).

Table 2. Project problem profiler (Appendix) structure

Characteristic Project Problems, Risk Factors	Categorization (Nominal)	Typical NPD Embedded SW	Typical IMPACT	Project STATUS	Project index
Program/Project Management					
Ineffective project management	Company	-	Critical!	x_1	y_1
Inadequate planning and task identification	Project	-	Moderate	x_2	y_2
Inter-component or inter-group dependencies	Project	NPD special concern!	Major	x_3	y_3
Personnel Management					
…					

The matrix has two main sections. The static part is basically a directory of typical software project problem factors, with a special emphasis on NPD embedded software projects. It comprises the following read-only fields (see Appendix):

- *Characteristic Project Problems, Risk Factors*:
  This column is a list of potential problem factors. They are grouped under the main sections of Program/project Management, Personnel Management, Scheduling and Timing, Requirements Management, System Functionality, Resource Usage and Performance, and Subcontracting. Under these main headings there are two levels of subgroups (only level 1 shown in Table 2).

- *Categorization (Nominal)*
  The problem items are further categorized according to the scope (Business Milieu / Company / Project / Team / Individual), class (Development Environment / Product Engineering / Program Constraints), type (Business / Technical / Process / People / Organizational), and the project phase of most likely concern (Project Initialization / Scoping / Planning / Execution / Completion).
- *Typical NPD Embedded SW*
  This highlights those problem areas, which are typically of special significance in embedded software projects (see "NPD special concern!" in Table 2).
- *Typical IMPACT*
  This value indicates the typical seriousness (Critical-Major-Moderate) of the problem for the project success.

The latter part of the matrix is dynamic, intended to be filled in by the user (more about that in Ch. 3.2). It consists of the following two fields:

- *Project STATUS*
  This value is the current evaluation of the project status with respect to the problem items (No problem / Minor issue / Concern / Serious!).
- *Project INDEX*
  The project's profile is indicated as a numeric value for each problem item. It is calculated based on the fields *Typical IMPACT* and *Project STATUS* as defined below (Formula 1). This index can further be used to plot graphical profiles of the current project situation (Ch. 3.2).

The matrix has in principle been composed as follows. The reasoning is discussed further in Ch. 4.

We have distilled a wide range of typical project problem factors (*Characteristic Project Problems, Risk Factors*) based on the literature survey (Ch. 2), coupled with our own real-life product development project experiences, with a special focus on NPD embedded software project concerns. Currently our matrix contains some 500 problem items organized in three levels (23 / 121 / 334 items, respectively). For example the following references have been used as the sources: [4-6, 12-19, 22, 27, 29-32, 34-36, 39, 42-48].

Most of the problem items are straightforward statements (e.g., "Poor communication"), but some of them are in a form of questions (like "Does management work to ensure that all customer factions are represented in decisions regarding functionality and operation?"). We have normally used the exact wording of the respective sources, with only some minor editorial changes.

The main grouping of the problem items is initially based on the seminal Boehm's risk list, refined by Ropponen and Lyytinen [5, 26]. We have in addition augmented it with one more main group: program/project management (comprising overall planning and coordination).

The problem item categorization (*Categorization (Nominal)*) is only suggestive. The *Scope* field is based on [21] and the *Class* field follows [31].

We have then estimated the relevance and typical impact of each problem item for NPD embedded software projects (*Typical NPD Embedded SW, Typical IMPACT*).

This evaluation is based partially on the ranking of the respective sources (if any given), and partially on our own experiences.

Finally, the *Project INDEX* is calculated according to the following formula:

$$Project\ INDEX_i = Weight * Typical\ IMPACT_i * Project\ STATUS_i \quad (1)$$

where the scales are currently defined as follows:

Weight:	1 (constant)
Typical IMPACT:	0-3 (Critical = 3)
Project STATUS:	0-3 (Serious = 3)
Project INDEX:	0-9

This formula is influenced by the commonly used calculation rule of risk exposure (more in Ch. 4.3).

3.2 Using the Profiler

The profiler matrix (Appendix) is in principle intended to be used as follows:

- For each problem item (level 1, 23 items altogether):
  - Answer the following question:
    - *Is this currently a problem in our case?*
    - If so, how serious is it (Minor issue / Concern / Serious)?
  - Write your rating down to the corresponding cell of the matrix (x_i in Table 2).
  - The corresponding *Project INDEX* value can then be calculated (y_i in Table 2).
- Finally, the *Project INDEX* values can be plotted graphically like illustrated in Appendix (Profile Chart). This gives a visual profile of the project's problem situation. The results can now be utilized in various ways during the course of the project (see Ch. 3.3).

For helping the evaluation of each main level (1) problem items, the lower-level (2, 3) items of the matrix can be used as guidance of thinking. For example, under the problem heading "New market with uncertain needs", there are more detailed items as illustrated in Appendix (Problem Sheet). The user can first ponder these lower-level items (at least part of them), and then give the aggregate rating of the level 1 item accordingly.

Naturally one can utilize the matrix also partially for example in case some sections are irrelevant (e.g., Subcontracting). On the other hand, it is of course also possible to extend the matrix with new problem items.

We have implemented the matrix as a computerized spreadsheet, which makes it easy to browse the different levels of the problem items, and automate the *Project INDEX* calculations and plottings. The Search functions of the spreadsheet can be used for example to find all problem items with certain keywords (e.g., "NPD").

3.3 Application Possibilities

The profiler matrix (Appendix) is a versatile tool. There is no one right way of using it. However, our key idea is to utilize it as follows:

- The project manager can use the matrix to self-assess her project (even privately). This assessment can be done while preparing the initial project plan as well as periodically during the course of the project:
  - The initial evaluation gives early insight and warning.
  - During the course of the project, the project manager can use the problem profile to focus the management activities on the alarming areas and trends.
  - The problem matrix can also be used as a tool in project (or iteration) post-mortem reviews. What were the biggest problems? The profile data could then be utilized for future projects (or iterations) for reference purposes.
- The assessments can also be done as group exercises together with the project team. The project manager and the project team could compare their evaluations.
- A more objective assessment ("health check") could be done by an outsider expert (such as a Quality Manager). The program and even corporate management could further utilize such information for ranking the individual projects. This kind of a ranking of risky projects have been investigated in [49]. This may be sensitive.

Naturally it is not enough to just recognize the problems. The project manager has to use other means to link the current identified problems to consequent improvement actions. In some cases no immediate action may be needed, while in other areas alarming trends (e.g., constant flow of unreasonable requirements changes) may require improvements even external to the current project. Combined results of individual project assessments could also be used for larger-scale company process improvement purposes (e.g., portfolio management).

4 Evaluation and Discussion

4.1 Empirical Experiments

We have conducted some empirical experiments with the problem profiler matrix (Appendix) in certain industrial NPD project environments at a large company developing telecommunications products containing embedded software. The method was to let the project managers to assess their project status with the matrix. Based on the responses, we expected to be able to draw conclusions about how well the profiler captures real project problem situations.

The following project background information was first recorded:

- product type: terminal / network element / etc.
- project nature: new features / completely new product / platform development
- project size, length (order of magnitude)
- major dependencies (e.g., hardware development, system integration)
- current state: launch / active / ending / completed / canceled

The project managers were then asked to fill in the problem matrix like instructed in Ch. 3.2. The survey was conducted by e-mail.

Table 3 shows a quantitative summary of the responses provided by the project manager (or the project quality manager). For confidentiality reasons the actual problem profile values cannot be shown here. In these project cases 5 common problem items (out of 23, level 1) were identified. All respondents provided additional narrative description of their project's main issues. This data was not codified, however.

Table 3. NPD project case studies

	Project Case	# of Problem Items flagged (out of 23)	# of Problem Items assessed as 'Serious!'	# of 'NPD special concern' items (out of 6)
1	Terminal software platform subsystem, new features; Project ending.	8	2	2
2	Network element software, completely new product; Project completed.	17	5	6

We can see that the profiler matrix captured critical problem areas of the case study NPD projects. None of the project cases identified any such significant problems that were not covered by the matrix. It is not possible to say, if the matrix approach highlighted such problem areas which had not yet been seen by the project manager.

4.2 Answering the Research Questions

We have composed a structured directory of typical problems encountered in NPD embedded software projects. This matrix (Appendix) helps identifying the project problems by pointing out such key concern areas (Question 1 in Ch. 2.4). The matrix is certainly not an all-encompassing database of all possible problem items, but the idea is to guide the thinking like a checklist and a structured interview technique. The user is encouraged to consider further problem items.

There are many ways of using the matrix, as described in Ch. 3.3. It can thus be used to check the "health" of the embedded software projects either internally or independently by an outsider assessor (Question 2 in Ch. 2.4). Naturally such checking can only give partial suggestions of the status of the project, but if this assessment indicates even some problems, further focused investigations should be considered. On the other hand, if there seem to be only very few problems (even none at all), one should become equally suspicious.

The matrix (Appendix) is composed with a generic viewpoint of NPD projects. While utilizing it in actual projects, it is important to understand the overall positioning and the nature of the project. Two such major issues are the front-end activities done prior to starting the actual software development project, and the level of new technology development involved. In NPD projects it is equally important to consider both commercial as well as technical risks [42, 46/Ch. 12, 50].

4.3 Limitations

We acknowledge the following limiting factors and constraints of our propositions presented in Ch. 3:

- The prescribed problems items scoping and categorization of the problem matrix (Appendix) are inherent bias factors. That could possibly skew the project's problem space exploration (even subconsciously). In some cases the assessor has to make a subjective mental mapping between her actual problems and the ones written in the matrix – unless there is an exact match. Consequently, different projects could show somehow different profiles, although the underlying problems would really be the same. These are typical pitfalls with checklist-based approaches [7].
- It is not reasonable to attempt to compose a complete list of absolutely all the possible project problems. Our matrix (Appendix) should therefore not be taken as a universal answer to all questions but merely a framework of thought. The usefulness of the matrix depends much on the creativity, experience, and competence of the project manager.
- There are many ways of categorizing and grouping different problem items, and currently our matrix shows only one way of doing it. Some of the lowest-level problem items could have been consolidated, but we have chosen to keep them separate for reference purposes. However, it is important to realize, that many problem items could be grouped under multiple categories, and there are different levels of problems and cause-effect dependencies. Notably the computerized spreadsheet of the matrix (Appendix) makes it possible to reorganize the problem items and groupings quite easily.
- We have highlighted those problem areas, which are usually pivotal in industrial NPD environments (*Typical NPD Embedded SW*). However, this is to some extent relative to the actual project circumstances, and in some cases certain other areas could still be key concerns. There is no guarantee, that following the matrix will always reveal the most important project problems.
- We have given suggestive default values of the typical impacts of the different problems (*Typical IMPACT*). However, the actual severity may vary depending on the project situations. What is typically a "showstopper" in most cases may still be manageable in some projects – with extreme measures. In addition, the sum effect of different problem factors may amplify (or lessen) the actual impact. The *Typical IMPACT* values should thus – if necessary – be adjusted (calibrated) to ensure the fidelity of the calculated *Project INDEX*.
- The *Project INDEX* value is not an absolute measure of the project's status. It is merely a gauge of potential warning signals. In particular, it should not be used to rank different projects unless the same person has done the underlying evaluation according to equal criteria. The ultimate project success/failure cannot be determined based on this assessment alone (for example because of business factors).
- The suggested self-assessment method is obviously subjective. Healthy self-criticism is necessary in order to avoid delusion. Cross-checking with multiple assessors is therefore recommended like described in Ch. 3.3.

4.4 Discussion

The underlying theoretical foundation of our approach is in conventional project issue and risk management. What is said about risk identification is in general applicable here, too. However, we have taken a specific viewpoint of product development projects with embedded software concerns. While there is much related work published

about typical software project risks and failure factors in general (see Ch. 2.1), not many investigations focus on embedded software projects, and only very few take the NPD context into account. We see problem-awareness an inherent part of intelligent project management practice in turbulent NPD environments.

Our problem matrix (Appendix) is in addition a survey of the related literature, showing what different problem areas have been acknowledged by different investigations over the years. Some common areas are identified by many studies, while some problems are less frequently advocated, depending on the scope and viewpoints of the investigations. Our special focus of NPD embedded software projects is not often published.

The question of how to group the project problem factor space has been addressed by many investigations over the years. Clearly, there is no one absolutely right universal categorization, but it depends on the selected viewpoints. A notably rigorous approach is presented in [27]. Traditional general-purpose categorizations are available in standards and other project management guides (e.g., PMBOK, ISO/IEC 15504). We have selectively adopted them. One newer alternative has been proposed in [51]. A life-cycle process area categorization aimed specifically for embedded products development is proposed in [34]. Product integration is one typical key problem area. Note, however, that with a computerized tool it is not necessarily binding to fix any one particular grouping, but the user could basically reorganize the problem item space from different points of views.

There is a profound underlying difference of our project problem assessments and those ones done following general-purpose frameworks, such as CMMI. While such generic models suggest a set of key activities expected to be performed for good software engineering and management, our problem matrix (Appendix) does not prescribe any particular activities. For example, while requirements management is one of the level 2 key process areas in the CMMI model, we simply ask the project manager to evaluate, whether it is a problem or not in her case. Such situational problem diagnosis has been applied to embedded software projects in [52].

A high-level project risk factor matrix is shown in [53]. It includes some basic technology, product acceptance, and project execution risks. A weighting scale is suggested for each risk area. This is basically similar to our problem matrix.

One recent, similar to our questionnaire-based approach of recognizing 'risky' software projects is proposed in [54]. Likewise, they compose their questionnaire (having the main categories of requirements, estimations, planning, team organization, and project management) following a literature survey and some industrial experiences of embedded software projects. However, more detailed embedded software and NPD problem items are not covered.

A general-purpose (not limited to IT) project risk rating method has been presented in [49]. It is similar to our method in the sense that the project manager rates a set of project risk factors (risk drivers, e.g., novelty), and the overall project risk level is then calculated accordingly.

A project uncertainty profile is proposed in [55]. Overall business, product, project, and organizational risk factors are rated according to their level of uncertainty. This is in principle similar to our problem profiling technique.

A project assessment method in terms of overall complexity and uncertainty is proposed in [56]. Both complexity and uncertainty are rated based on a few

prescribed attributes (e.g., domain knowledge gaps, dependencies, project duration). Project complexity and uncertainty indices are then calculated. This is essentially a subset of our problem profile. However, in our case it is up to the project manager to evaluate whether the increased uncertainty caused for example by a long project duration is really a problem.

Some publicly available / commercial risk management software tools provide similar functionalities to our problem matrix. However, the purpose of our matrix is not to replace such tools.

5 Conclusions

We have constructed some pragmatic aids for understanding the various trouble spots of NPD embedded software projects. The outcome is not any particular solution for managing such projects, but it provides a holistic view over the problem space. A wise project manager can utilize this view for managing her particular project successfully even under unfavorable circumstances. After all, such cases are not so unusual in modern turbulent product development environments [48].

The problem matrix (Appendix) is certainly not a silver-bullet troubleshooter of every possible project problem case. However, the idea is to illuminate the overall picture of the project's problem space so that the major areas are revealed. Based on this guidance, the project manager can then focus on analyzing the problem indicators in more detail according to the project's actual contextual information. The usefulness of the matrix thus depends much on the experience of the project manager. For less experienced managers it shows the major areas to be considered to begin with. For a more experienced user, it serves merely as a structured checklist, giving hints and reminders of the typical trouble spots.

This paper leaves room for further study:

1. More empirical validation: At the time of the writing we are able to present only limited empirical case data about our propositions. More data should be collected by experimenting the matrix (Appendix) like described in Ch. 4.1. The empirical validation could follow the principles used in [54]. In particular, are there any significant problem areas that are currently not addressed in the matrix? How much does the prescribed categorization bias the problem assessments?
2. More rigorous categorization of the problem space.
3. As defined now, the calculated *Project INDEX* value is a simple measure with certain bias limitations (see Ch. 4.3). More advanced measures could possibly be developed for example by taking into account the basic nature of the project (e.g., high market uncertainty vs. high technological uncertainty). Can the overall project uncertainty and complexity be measured? Does the project type change it?
4. What can we say about projects based on their problem profiles (Appendix: Profile Chart)? Can we identify particularly risky (or "unhealthy") projects [49]? When should we cancel or not even start the project? How does the problem profile change over the project's life-cycle? A reference database of problem profiles of both successful and failed projects could be collected.

5. Problem-conscious project management: The problem matrix could be extended with suggestions of potential maneuvers for each problem item. We have already investigated elsewhere, how different software process models tackle certain project problems [57, 58]. Those results could be linked to the problem matrix.

Acknowledgements

The author would like to thank Maarit Laanti (Nokia Corporation) for her influence and critique. We are also grateful to the anonymous case study project managers.

References

1. Farbman White, S., Melhart, B.E., Lawson, H.W.: Engineering Computer-Based Systems: Meeting the Challenge. IEEE Computer **34**(11) (2001) 39-43
2. Iansiti, M.: Shooting the Rapids: Managing Product Development in Turbulent Environments. California Management Review **38**(1) (1995) 37-58
3. MacCormack, A., Verganti, R., Iansiti, M.: Developing Products on "Internet Time": The Anatomy of a Flexible Development Process. Management Science **47**(1) (2001) 133-150
4. Mullins, J.W., Sutherland, D.J. New Product Development in Rapidly Changing Markets: An Exploratory Study. Journal of Product Innovation Management **15** (1998) 224-236
5. Boehm, B.W.: Software Risk Management: Principles and Practices. IEEE Software **8**(1) (1991) 32-41
6. DeMarco, T., Lister, T.: Walzing with Bears: Managing Risks On Software Projects. Dorset House Publishing, New York (2003)
7. Kontio, J.: Software engineering risk management: a method, improvement framework, and empirical evaluation. Helsinki University of Technology (2001)
8. Glass, R.L.: Software Runaways. Prentice-Hall, Upper Saddle River (1998)
9. Pavlak, A.: Project Troubleshooting: Tiger Teams for Reactive Risk Management. Project Management Journal **35**(4) (2004) 5-14
10. Kwak, Y.H., Stoddard, J.: Project risk management: lessons learned from software development environment. Technovation **24** (2004) 915-920
11. Sheremata, W.A.: Finding and solving problems in software new product development. Journal of Product Innovation Management **19** (2002) 144-158
12. Conrow, E.H., Shishido, P.S.: Implementing Risk Management on Software Intensive Projects. IEEE Software **14**(3) (1997) 83-89
13. Evans, M.W., Abela, A.M., Belz, T. Seven Characteristics of Dysfunctional Software Projects. CrossTalk **15**(4) (2002) 16-20
14. Houston, D.: Results of Survey on Potential Effects of Major Software Development Risk Factors. http://www.eas.asu.edu/~sdm/dhouston/risksrvy.htm (1999) (accessed February 2005)
15. Jones, C.: Patterns of Software System Failure and Success. International Thompson Computer Press, Boston (1996)
16. May, L.J.: Major Causes of Software Project Failures. CrossTalk **11**(7) (1998) 9-12
17. Reel, J.S.: Critical Success Factors In Software Projects. IEEE Software **16**(3) (1999) 18-23
18. Reifer, D.: Ten Deadly Risks in Internet and Intranet Software Development. IEEE Software **19**(2) (2002) 12-14

19. Wiegers, K.E.: Know Your Enemy: Software Risk Management. http://www.processimpact.com/articles/risk_mgmt.pdf (1998) (accessed February 2005).
20. Brooks, F.P. Jr.: The Mythical Man-Month: Essays on Software Engineering (20th Anniversary Edition). Addison-Wesley (1995)
21. Curtis, B., Krasner, H., Iscoe, N.: A Field Study of the Software Design Process for Large Systems. Communications of the ACM **31**(11) (1988) 1268-1287
22. McConnell, S.: Rapid Development: Taming Wild Software Schedules. Microsoft Press, Redmond (1996)
23. McConnell, S.: Software Project Survival Guide. Microsoft Press, Redmond (1998)
24. Royce, W.: Software Project Management. Addison-Wesley (1998)
25. Brown, S.L., Eisenhardt, K.M.: Product Development: Past Research, Present Findings, and Future Directions. Academy of Management Review **20**(2) (1995) 343-378
26. Ropponen, J., Lyytinen, K.: Components of Software Development Risk: How to Address Them? A Project Manager Survey. IEEE Trans. Software Engineering **26**(2) (2000) 98-111
27. Schmidt, R., Lyytinen, K., Keil, M., Cule, P.: Identifying Software Project Risks: An International Delphi Study. Journal of Management Information Systems **17**(4) (2001) (Spring) 5-36
28. Smith, J.M.: Troubled IT Projects – prevention and turnaround. IEE (2001)
29. May, G., Ould, M.: Software project casualty. IEE Engineering Management Journal **12**(2) (2002) 83-90
30. Fairley, R.E., Willshire, M.J.: Why the Vasa Sank: 10 Problems and Some Antidotes for Software Projects. IEEE Software **20**(2) (2003) 18-25
31. Carr, M., Kondra, S., Monarch, I., Ulrich, F., Walker, C.: Taxonomy-Based Risk Identification (Technical Report CMU/SEI-93-TR-6). SEI (1993)
32. Brown, W.J., McCormick H.W. III, Thomas, S.W.: AntiPatterns in Project Management. John Wiley & Sons, New York (2000)
33. Ould, M.A.: Managing Software Quality and Business Risk. John Wiley & Sons, Chichester (1999)
34. Kuvaja, P., Maansaari, J., Seppänen, V., Taramaa, J.: Specific Requirements for Assessing Embedded Product Development. In: Proc. International Conference on Product Focused Software Process Improvement (PROFES) (1999) 68-85
35. Rauscher, T.G., Smith, P.G.: Time-Driven Development of Software in Manufactured Goods. Journal of Product Innovation Management **12** (1995) 186-199
36. Ronkainen, J., Abrahamsson, P.: Software development under stringent hardware constraints: Do agile methods have a chance? In: Proc. 4th Int'l Conf. Extreme Programming and Agile Processes in Software Engineering (2003) 73-79
37. Kettunen, P.: Managing embedded software project team knowledge. IEE Proc. – Software **150**(6) (2003) 359-366
38. Riek, R.F.: From experience: Capturing hard-won NPD lessons in checklists. Journal of Product Innovation Management **18** (2001) 301-313
39. Wheelwright, S.C., Clark, K.B.: Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency, and Quality. The Free Press, New York (1992)
40. Song, X.M., Montoya-Weiss, M.M.: Critical Development Activities for Really New versus Incremental Products. Journal of Product Innovation Management **15** (1998) 124-135
41. Ernst, H.: Success factors of new product development: a review of the empirical literature. International Journal of Management Reviews **4**(1) (2002) 1-40
42. Cooper, R.G., Edgett, S.J., Kleinschmidt, E.J., Benchmarking Best NPD Practices – III. Research • Technology Management **47**(6) (2004) 43-55

43. Jones, C.: Minimizing the Risks of Software Development. Cutter IT Journal **11**(6) (1998) 13-21
44. Jones, C.: Software Assessments, Benchmarks, and Best Practices. Addison-Wesley (2000)
45. Rautiainen, K., Lassenius, C., Nihtilä, J., Sulonen, R.: Key Issues in New Product Development Controllability Improvement – Lessons Learned from European High-tech Industries. In: Proc. Portland Int'l Conf. Management of Engineering and Technology (PICMET) (1999)
46. Smith, P.G., Reinertsen, D.G.: Developing Products in Half the Time: New Rules, New Tools. John Wiley & Sons, New York (1998)
47. Ulrich, K.T., Eppinger, S.D.: Product Design and Development. McGraw-Hill, New York (2000)
48. Yourdon, E.: Death March – The Complete Software Developer's Guide to Surviving "Mission Impossible" Projects. Prentice-Hall, Upper Saddle River (1999)
49. Baccarini, D., Archer, R.: The risk ranking of projects: a methodology. International Journal of Project Management **19** (2001) 139-145
50. Holmes, M.F., Campbell R.B. Jr.: Product Development Processes: Three Vectors of Improvement. Research • Technology Management **47**(4) (2004) 47-55
51. Keil, M., Cule, P.E., Lyytinen, K., Schmidt, R.C.: A Framework for Identifying Software Project Risks. Communications of the ACM **41**(11) (1998) 76-83
52. Iversen, J., Nielsen, P.A., Nørbjerg, J.: Situated Assessment of Problems in Software Development. The DATA BASE for Advances in Information Systems **30**(2) (1999) (Spring) 66-81
53. Fitzgerald, D.: Principle-Centered Agile Project Portfolio Management. Agile Project Management Advisory Service Executive Report 6(5), http://www.cutter.com/project/fulltext/reports/2005/05/index.html (2005) (accessed June 2005)
54. Takagi, Y., Mizuno, O., Kikuno, T.: An Empirical Approach to Characterizing Risky Software Projects Based on Logistic Regression Analysis. Empirical Software Engineering **10** (2005) 495-515
55. DeCarlo, D.: Leading Extreme Projects to Success. Agile Project Management Advisory Service Executive Report 5(8), http://www.cutter.com/project/fulltext/reports/2004/08/index.html (2004) (accessed June 2005)
56. Little, T., Greene, F., Phillips, T., Pilger, R., Poldervaart, R.: Adaptive Agility. In: Proc. Agile Development Conference (ADC) (2004) 63-70
57. Kettunen, P., Laanti, M.: How to steer an embedded software project: tactics for selecting the software process model. Information and Software Technology **47**(9) (2005) 587-608
58. Kettunen, P., Laanti, M.: How to Steer an Embedded Software Project: Tactics for Selecting Agile Software Process Models. In: Proc. International Conference on Agility (ICAM) (2005) 241-257

Appendix. Project Problem Profiler Matrix

Problem Sheet: The following shows the complete level 1 table. The *Project STATUS* values put here do not represent any particular project case, but in our experience this kind of ratings could well be observed in typical NPD embedded software projects.

Characteristic / Project Problems, Risk Factors	Categorization (Nominal) SCOPE	CLASS	TYPE	PHASE	Typical NPD Embedded SW	Typical IMPACT	Project STATUS	Project INDEX
Program/Project Management								
Ineffective project management (multiple levels possible)	Company	Development Envir	Organizational	Project ALL	~	Critical!	No problem	0
Inadequate planning and task identification	Project	Development Envir	Process	Project Planning	~	Moderate	No problem	0
Inter-component or inter-group dependencies	Project	Development Envir	Process	Project Execution	NPD special concern!	Major	No problem	0
Personnel Management								
Personnel shortfalls (lack of qualified personnel and their	Company	Program Constrair	Organizational	Project ALL	~	Major	Minor issue	2
Inability to acquire resources with critical skills	Project	Development Envir	Organizational	Project Init	~	Critical!	No problem	0
Instability and lack of continuity in project staffing	Project	Program Constrair	Organizational	Project ALL	~	Major	Minor issue	2
Lack of staff commitment, low morale	Team	Development Envir	People	Project ALL	~	Major	No problem	0
Scheduling and Timing								
Unrealistic schedules, budgets (time and budget estimated	Company	Program Constrair	Business	Project Planning	NPD special concern!	Critical!	Minor issue	3
Inherent schedule flaw	Project	Development Envir	Process	Project Planning	~	Major	No problem	0
Inaccurate cost estimating	Project	Development Envir	Process	Project Planning	~	Moderate	No problem	0
Poor productivity	Project	Development Envir	Process	Project Execution	~	Major	No problem	0
Requirements Management								
New market with uncertain needs	Business Milieu	Program Constrair	Business	Project Scoping	NPD special concern!	Critical!	No problem	0
Gold plating (adding unnecessary features)	Project	Product Engineerir	Process	Project Scoping	~	Moderate	Minor issue	1
Continuing stream of requirements changes (uncontrolled	Project	Product Engineerir	Process	Project Scoping	~	Major	Concern	4
System Functionality								
Complex application	Company	Product Engineerir	Technical	Project Planning	~	Major	Minor issue	2
Developing wrong software functions (functions that are not	Project	Product Engineerir	Technical	Project Scoping	NPD special concern!	Critical!	No problem	0
Specification breakdown	Project	Product Engineerir	Process	Project Execution	~	Major	No problem	0
Developing wrong user interface (inadequate or difficult)	Project	Product Engineerir	Technical	Project Execution	~	Moderate	No problem	0
Resource Usage and Performance								
Real-time performance shortfalls	Team	Product Engineerir	Technical	Project Execution	NPD special concern!	Major	Serious!	6
Ineffective development technologies	Team	Development Envir	Process	Project Planning	~	Moderate	No problem	0
Straining computer science capabilities (lacking technical	Team	Product Engineerir	Technical	Project Execution	NPD special concern!	Critical!	No problem	0
Subcontracting								
Shortfalls of externally furnished components (poor quality)	Project	Program Constrair	Process	Project Execution	~	Moderate	No problem	0
Shortfalls of externally performed tasks (poor quality or	Project	Program Constrair	Process	Project Execution	~	Moderate	No problem	0

Problem Sheet: The following shows a section of the expanded level 2-3 table (c.f., above).

Characteristic Project Problems, Risk Factors	Categorization (Nominal) SCOPE	 CLASS	 TYPE	 PHASE	Typical NPD Embedded SW	Typical IMPACT		Project STATUS	Project INDEX
Requirements Management									
New market with uncertain needs	Business Milieu	Program Constrair	Business	Project Scoping	NPD special concern!	Critical!		No problem	0
Lack of clear product vision					NPD special concern!	Critical!	*No problem*	⮥	
New product strategy exists					NPD special concern!	Major	-	⮥	
Misalignment with business goals					~	~	-	⮥	
Business needs change					~	~	-	⮥	
What market segments should be considered in					~	~	-	⮥	
Who is the target customer?					NPD special concern!	Critical!	-	⮥	
Inadequate coverage of target markets with competitive					~	~	-	⮥	
Is the concept chosen by the team best suited to the					NPD special concern!	Major	-	⮥	
Focus on current customers and confusion about future					~	~	-	⮥	
Is the product a winner?					NPD special concern!	Critical!	-	⮥	
Product advantage: unique, superior					~	~	-	⮥	
Lack of Product Distinctiveness (new product not as					~	~	-	⮥	
Poor timing of market introductions of products					NPD special concern!	Major	*No problem*	⮥	
Uncertainty associated with the inability of the customers to					NPD special concern!	Critical!	*No problem*	⮥	
Sharp, early product definition (fact-based, e.g., the benefits					~	~	*No problem*	⮥	
Failure to gain user commitment: Laying blame for "lack of					~	~	*No problem*	⮥	
Customer-furnished items or information					~	~	*No problem*	⮥	
Lack of client support					~	~	*No problem*	⮥	
Lack of contact person's competence					~	~	*No problem*	⮥	
Gold plating (adding unnecessary features)	Project	Product Engineerir	Process	Project Scoping	~	Moderate		Minor issue	1
Continuing stream of requirements changes (uncontrolled	Project	Product Engineerir	Process	Project Scoping	~	Major		Concern	4

Profile Chart: The following shows an example plot of the problem profile chart based on the sample Problem Sheet values above.