TRADE-OFF'S IN WEB APPLICATION DEVELOPMENT
How to Assess a Trade-off Opportunity

Sven Ziemer and Tor Stålhane
Department of Computer and Information Science, Norwegian University of Technology and Science, Sem Sælandsvei 7-9, NO-7491 Trondheim, Norway

Keywords: Trade-off, decision making.

Abstract: This paper discusses a trade-off strategy for small projects and presents a preliminary guidelines for assessing the appropriateness of a trade-off. The motivation of this work is to make the development team aware of the performed trade-offs and to see both the associated opportunities and risks, to assess the appropriateness of a trade-off and to gather information for the future development of the system. The development environment of a web application is changing over time and it is important to know what the success criteria are at any time.

1 INTRODUCTION

Web Applications have become part of our every day life in such a way that we expect them to work properly and that we can rely on them, both with respect to their availability and our satisfaction. Hence, the quality of web applications is of interest to all users, that want reliable access to the services offered by the web applications of their choice such as internet banking, online shopping, news services and entertainment. Improving the development process to build high quality applications that fulfil the end-users expectation plays a key role in keeping the users satisfied. At the same time, businesses of all sizes and kinds are relying on their web applications to promote their services and to fulfill the needs and expectations of their customers. They too are interested in the quality of the applications they are relying on, but they are also focused on their timely delivery. Many web applications have to compete for their users and customers against other providers of a similar service. Being the first web site to offer new functionality, or being the site with the most complete functionality for a user group, is important in this competition. In other cases, web applications provide their services to support events and have to meet tight deadlines, so that the event can be promoted in the best possible way.

Developing web applications in competitive environments adds to the complexity of balancing functional and non-functional requirements by adding the time to market requirement. Finding this balance involves performing one or several trade-offs, such as architectural trade-offs or security trade-offs. A different class of trade-off's involves development activities and tasks that are applied to build an application. This trade-off is about the emphasize that is placed on the steps involved in building applications, and with what rigour and level of detail they are performed.

In a series of interviews that we have conducted with seven Norwegian companies, we studied both the companies' development practises in general and what type of trade-offs the companies performed and what trade-off strategies that were applied in particular (Ziemer and Stålhane, 2006). We found that the companies tend to specify their requirements in informal ways, depending on the experience of the developers and their domain understanding. Performing a trade-off with such practices is not easy since it is hard to assess the consequences that a potential choice for a given requirement might have on other important requirements. There are no hard numbers that can be used in prediction or estimation models that can be compared against measured real values at a later stage. The only knowledge available is the qualitative knowledge stemming from expert judgements and the stakeholders opinions and beliefs.

Part of performing a trade-offs is to share a common understanding of the stakeholders priorities and

Ziemer S. and Stalhane T. (2007).
TRADE-OFF'S IN WEB APPLICATION DEVELOPMENT - How to Assess a Trade-off Opportunity.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Internet Technology*, pages 267-274
DOI: 10.5220/0001286702670274

interests, and an understanding of the consequences these interests and needs have for the the application. When web applications are developed with the aforementioned development practices, there is a need for an approach to perform trade-offs that work with these practices and that exploits the available knowledge.

In this paper we will present an approach to assess the consequences of a trade-off in an environment with informal and tacit knowledge. This is done with the available qualitative information. The rest of this paper is organised as follows: some characteristics of web application development are presented in section 2. Next, a number of trade-off opportunities will be explored in section 3, and some preliminary guidelines for performing trade-offs are presented in section 4. An small example is presented in section 5. Finally, a discussion and conclusions are is presented in section 6.

2 WEB APPLICATION DEVELOPMENT

Web application development is characterized – among many other factors – by the informality of its development practices, the available knowledge and by an ever changing development process. These factors have an influence on how trade-offs can be performed in web application development. In our survey of seven web developing companies (Ziemer and Stålhane, 2006) the companies did not have a trade-off strategy and were in many cases not aware that they performed trade-offs.

Not being aware of performing trade-offs means that the freedom of choice is given away. This does not imply that the trade-off was a bad one – discussing what a good or appropriate trade-off is, is a seperate discussion – but the freedom to make a different choice is given away. This could be a choice that will enable the future success of the application, or a choice that will lead the future development into a certain direction. Any decision in based on a number of assumptions. To make a good decision the assumption have to be correct and they have to be the right set of assumptions to find the best decision. What is needed is a trade-off strategy that helps developers to assess the appropriateness of a trade-off – and the associated opportunities and risks – in a given situation or context. A trade-off results in a decision that can influence project and product risks and the product strategy. A trade-off strategy should make performing a trade-off an intentionally choice. This means that the pro and cons of a possible trade-off have to be assessed and that the stakeholders have some expectations with respect to the result of that trade-off. Any engineered system has to live with the consequences of the decisions made during its design and construction. Whether a trade-off will yield the expected result or not can only be seen later. The decisions made for a system will restrict the future development of the system in such a way that it will be troublesome to change a system in a way that is not foreseen or that is in conflict with the underlying assumptions of the decisions made so far (see (Lago and van Vliet, 2005)). Therefore, it is important to perform these trade-offs explicitly, so that the information a trade-off is based on, is documented for future use.

2.1 Informal Knowledge

Developing software is a knowledge intensive endeavour. To build a software system that satisfies its stakeholders it is necessary to understand not only what functions the system shall perform, but also the need or motivation for this functionality, the context in with this functionality will be used, etc. This knowledge is found in existing systems, processes or practices as well in individual, institutions and material structures (Hanseth, 2004).

The activities in a software process aim at finding this knowledge, elaborating on it and documenting it. This requires that the documented knowledge is unambiguous, so that it will have the same meaning to every stakeholder. Learning and conveying this knowledge to other members of a community is a difficult task. The knowledge-sharing model – also called the ”tacit-explicit model” – proposed by (Nonaka and Takeuchi, 1995) portrays this problem as a continuous cycle. This cycle consists of socialisation, externalisation, combination and internalisation.

Looking to the practices found in web application development (Ziemer and Stålhane, 2006), we see that a lot of knowledge is informal. Similar findings can be found in other studies, too. One study found that the focus in web development is on the functional requirements. Non-functional requirements or quality attributes are not specified with any rigor, partly because neither the developers nor other stakeholders are aware of them or tend to specify them as functional requirements based on previously experience (N. Yusop and Lowe, 2006). It seems that when TTM matters, also functional requirements are handled quite informally, by using analogies or examples.

The result of these practices is that the knowledge is not documented unambiguous and hence, that it can not be shared easily (Shull et al., 2004). This works only with experienced developers who have a good

understanding of the domain of the web application. There are other consequences to these practices:

- The knowledge about an application is fragmented. Every stakeholder possesses a piece of the knowledge about an application. Documenting this knowledge means bringing it together and looking at the system from several viewpoints. This is, however, not easy when the knowledge is fragmented.
- The knowledge is not unambiguous. It is open to interpretation from every stakeholder, who tends to look at it with his own goals and needs in mind. When discussing future aspects of an application, a common understanding is not necessarily present and have to be provided.
- When assessing the consequences of potential decisions, it is not possible to perform a formal analysis that investigates several aspects of a decision with some rigour.

This is not to say that it is wrong to develop applications with these development practices. Web applications *are* developed in this way and many web applications are quite successful. Still, it is important to be aware of that there is a lot of tacit knowledge involved. This is especially true when it comes to performing trade-offs.

2.2 Several Phases in the Life of a Web System

Web systems seem to go through several phases during their lifetime. There are several development phases, but there seem also to be several phases with respect to the objective and purpose of a web system in a competitive marketplace. The transition from one to an other phase is a continous one. We present three phases, that are not necessary distinct. They are rather snapshots at three different moments:

- The start-up phase. The web-page's main purpose is to catch attention. Quite often it is just a brochure page, showing some info such as articles for sale. Such pages are also often used to see if it is possible to catch the attention of the market place. Often they contain simple games etc. A useful concept here is "Attention is everything – quality is nothing".
- The mid-life phase. We have survived the fight for attention and have added more feature to our system. The system is now important for our business although we could probably survive without it.
- The final phase. The system has grown and is now critical for our business. Most of our marketing effort and a considerable part of our transactions are now conducted via our web-pages. A long period – e.g. more than a week – where the web-page could not be accessed could be fatal for our company.

These three phases and where we are in this picture are important since it will decide the risk we take when we change something. During the start-up phase the risk is low. We do not need any formal processes, neither for changes nor for change management. Trade-offs can be frequent and done just to see how they turn out. The decision process called the garbage-can process (Cohen et al., 1972) will do in most cases.

Things start to get more serious in the mid-life phase. Our web-page now generates a considerable amount of business. This implies that bad changes can have serious consequences. We thus need a better control over the changes, even though there always is the possibility to roll back to the previous version – at lest as a stop-gap action. Even so, a trade-off should now be done with more considerations to possible bad effects. In addition to a well-defined trade-off process we should now also consider possible bad effects – their probability and consequence together with possible barriers. The risk assessment can, however, still be done at a rather informal level.

The option to do a roll-back in case of a bad decision is dependent on a quick market feedback. Most of the time, the only market feedback we get is that our customers take their business elsewhere and when we discover this, it may be too late to do anything about it. It may be possible to avoid this problem by performing frequent user satisfaction surveys.

All the problems that we can run into in the mid-life phase will also be important in the final phase – only more so. The risk is greater, not because the uncertainty of the outcome has changed but because the potential loss has increased. We need a formal trade-off process, coupled with a formal risk assessment process.

2.3 Development Process Evolution

The change from one life phase to another is not without consequences for the applied development process. In order to face new risks that are introduced by this change, the development process has to adapt, either by introducing new activities or by changing existing activities to handle the risk. The same is also true for the opportunities that are associated with the risk. To exploit these opportunities the process has

to evolve. The authors of (Ramesh et al., 2002) state that this evolution of a development process will require different project cultures at different times.

Many companies developing web applications in a competitive environment have found their own success criterion. This can either be a technical or a non-technical factor. They apply development practises that support this success criterion and often establish an ad-hoc development process. As web applications change over times this can also change the success criterion. To stay in front in this competition the success criterion has to evolve, too. This makes an assessment of the success criterion and the present lifetime phase – with its risks and opportunities – necessary.

This process evolution does also effect the trade-offs and when it is appropriate to perform a trade-off or not. What was a good trade-off in an early lifetime phase can be a non-appropriate trade-off in a later phase. Also here, an assessment of the trade-off versus the present lifetime phase is advisable.

2.4 Assessment of Risks and Opportunities

When discussing risk and consequences, it is all too easy to forget that changes can also bring opportunities. It is the opportunities that drive the changes in web applications, not the risks. Thus, there must be a reasonable relationship between opportunities and risks. This implies that we need to focus on three factors:

- The opportunities – what are we trying to achieve with our changes? This implies that we need to identify opportunities and enablers.
- The risks – what can go wrong? This implies that we need to identify risks and barriers.
- The trade-offs – how can we balance the opportunities so that we choose the best combination?

The first two factors can be performed by doing a SWOT – Strength, Weakness, Opportunity and Threats – analysis. The last one – trade-offs – can be done as suggested in (Ziemer et al., 2005).

Risks in our case stem from two sources – how much do your company depend on a properly working web system and what will be the reaction from your current customers. These two sources are strongly interdependent and are both related to your customers' cost of moving to another supplier. Thus, knowing the current phase in a web system's life is not enough when we want to understand the risk of the system's owner. If the customer incurs large costs if he wants to move to another supplier, the risk related to changes is small. This is for instance the case if you are a bank or the homepage of apopular tourist destination. If you, on the other hand, sell a standard commodity in competition with many other suppliers, your risk is large. You will receive no complaints and no warnings; you will just loose your customers at an alarming speed. By the time you understand that your latest release was a bomb, it is probably way too late for roll back or error correction.

The other important factor is your attitude to risk or you general risk handling strategy. We observe two reactions to identified risks:

- This action, although it offers great opportunities, has so much risk related to it that we cannot do it.
- This action offers great opportunities. It has some risks related to it but since the activity is important we will start looking for ways to prevent or mitigate these risks.

Both of these risk handling strategies are possible, but only the second one is viable in the long run. The first one will inevitably lead to stagnation. Although it might be reasonable to assume that small companies would use the first alternative while small companies would use the second one, no such correlation has been observed.

In order to handle risk in a sensible way we need to do two things – we must:

- Understand our situation in the market – what will dissatisfied customers do – and which phase are we in concerning the web system's life – how much do we depend on our web system?
- Clarify our attitude to risky undertakings – avoidance versus control.

There exist standard approaches that can be used in order to assess risk / barriers and opportunities / enablers. Our main message here is that you need to consider these factors if you want to perform a sensible tradeoff analysis.

The result of the risk and opportunity analysis will, in our case, be a decision related to the choice of process. Low risk implies the need for little or no formal process and a short TTM while a high risk implies a formal process and a correspondingly long TTM.

2.5 Trade-off Strategy

Performing a trade-off – whether we are aware of it or not – will have a lasting effect on the web application and can restrict future possibilities. As web applications get larger and integrate an increasing number of other systems, they also become harder to change. Small web applications can be re-engineered within

short time. This is not the case for larger web applications. Thus, making the right decisions and performing good trade-offs is an important task.

It has already been mentioned that developing web applications with emphasise on TTM involves a lot of tacit knowledge, that can be expressed as expert judgements and beliefs. Having a trade-off strategy that guides the development team in performing trade-offs from iteration to iteration is helpful since it

- enables and supports the accumulation of explicit knowledge,
- helps to evaluate the results of trade-off decisions by tying together the decision and its consequences, thus enabling learning,
- keeps future options open by avoiding decisions that will prevent them.

By making this process explicit, the developers and the other stakeholders are aware of their decisions and they can thus trace the results back to their decision. They can formulate some future objective or goal for the applications and then steer the further development into this direction. When they have to face situations where it is not possible to make a decision into the preferred direction they will be aware of this, and the goal for the applications can – if necessary – be restated.

It makes no sense to talk about right trade-off versus wrong trade-offs. A trade-off represents a choice, and a trade-off was good or wise with respect to a stated goal or objective for the application. This involves in most cases the reconciliation of conflicting interest among the stakeholders. A second conflict that needs reconciliation is the one between short time and long time term objectives. Therefore, a trade-off strategy should include a way to

- reach a decision, that can be supported by all stakeholders,
- weight the consequences of a decision with respect to some stated overall goals or objectives.

In this way, it will be possible to make a decision that is as informed as possible – given that a lot of information is informal and tacit – and to evaluate the result of the decisions, and, when needed, to make the necessary corrections.

3 TRADE-OFF OPPORTUNITIES

The objectives of the Websys project is to study at trade-off's between time-to-market and software quality. 3 examples of development practice trade-offs' are presented in this section.

3.1 Requirement Specification

Requirement specification trade-off has been observed in companies (Ziemer and Stålhane, 2006) as a trade-off between the level of detail in requirement specification and time-to-market. Requirements are specified informally, through oral communication only and sometimes based on anecdotes. When the level of detail in requirement specification is

- **increased**, the development process turns more formal, and a longer TTM is to be expected. A more detailed requirement specification enables both a more detailed test phase of the developed software system, and a systematical way to manage a wide stakeholder involvement into the requirement process.
- **decreased**, the development process turns less formal, and a shorter TTM is to be expected. Communication with other stakeholders is open to more misunderstandings as every part in such an discussion has his own interpretation of the requirements. Also, user involvement becomes more limited.

The underlying assumption of this trade-off is that the developer(s) have detailed domain knowledge and that verification of the requirements is not necessary.

3.2 Release Planning

Release planning can be used as a trade-off between time-to-market and the amount of functionality and its associated return of investment to the stakeholders. A system is deployed in a number of successive deployments, where each deployments consists of a set of functionality. When the number of deployments are

- **increased**, time-to-market for every deployment is decreasing. The end users perceived value of a new deployment may be limited. Every deployment has also an associated cost, as it may generate technical problems or a peak in the user feedback. On the other hand, a larger number of smaller deployments enables an earlier return of investment for the stakeholders and opens up for a more flexible reaction to changes in the market.
- **decreased**, time-to-market for every deployment will be longer. The end users perceived value of a new release may be higher, as more functionality is deployed, but it comes at a later time.

The underlying assumption of this trade-off is that the stakeholders agree to find a way of expressing the value of a release. Dependent on the information

available, the value can either be expressed as a belief (as in (Ziemer et al., 2006)), as some scale or as a monetary value (see (Biffl et al., 2006)).

3.3 Testing

When testing there is a trade-off between test coverage and time-to-market. When interviewing companies developing web applications, we found companies where the applications were tested by developers with their expectation as a test criterion, and deployed upon reaching this criterion. When the test coverage is

- **increased**, the confidence to the software system is increasing, both with respect to the proper implementation of the functional requirements and quality attributes. This will also result in a longer TTM. The number of user reported bugs and problems should be low, and this should have an positive impact on the user satisfaction.
- **decreased**, the confidence in the application is decreasing. The developers do not know the problems that might arise when deploying an application with a low test coverage.

The underlying assumption is that the developer expectation to the application is in touch with the main user base, and that users are reporting the bugs and problems that they encounter.

4 QUALITATIVE ASSESSMENTS OF TRADE-OFFS

With no detailed information on important quality factors and development activities available, we still have the opportunity to use the qualitative information that expresses the stakeholders expert judgement and beliefs. This information can be used to assess a trade-off situation, in order to find a balance that will satisfy most stakeholders.

Web Development is highly iterative. For every iteration it should be possible to improve the knowledge about an application. Quality factors can be measured and judgements and beliefs can be enhanced with quantitative data. Using qualitative assessments when no other information is available can help directing the efforts of collection more quantitative data.

There are several ways to collect and express qualitative information. In the example in this paper we are using the SWOT analysis (Hill and Westbrook, 1997). Another possible method is Impact Analysis

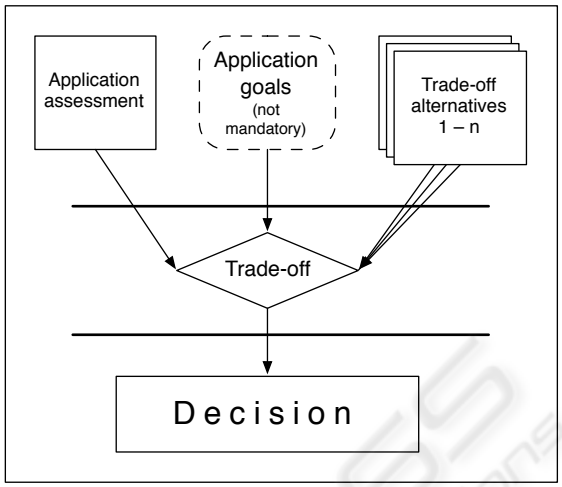

Figure 1: A simple trade-off model.

tables (Gilb, 2005). The most important thing, however, is to start an exchange of viewpoints, beliefs and judgements between all stakeholders or – if the number of stakeholders is too big – from of the most important stakeholders.

When performing a trade-off, the number of stakeholders involved is important. In many cases, it is preferable to use a small number of key stakeholders. In the context described in this paper, it is important to involve all stakeholders in order to elicit as much knowledge as possible. In order to find a proper balance it is crucial to have access to the whole picture, so that the trade-off can be performed based on this information.

Reconcilement of conflicts – such as conflicting requirements or conflicting priorities – is also an important issues. One approach is to weight the importance of the stakeholders and thereby making the opinion of some stakeholders more important then the opinion of other stakeholders. A second approach is to find a consensus between all stakeholders. This works only when the number of stakeholders is small. In a setting with a lot of tacit knowledge distributed among the stakeholders, this approach is well suited to get a common understanding of the trade-off situation. This way, all stakeholders have to listen to the opinions of the other stakeholders.

Performing a trade-off involves the following steps (see the simple trade-off model in figure 1):

- **Decide on a list of alternative decisions or choices** – Each potential alternative is described shortly. Make sure that each alternative is distinct. This is important in order to be able to assess the consequences. When using textual descriptions it will be hard to describe alternatives that are close

to each other in a distinctive way.

- **Assess each alternative** – This steps includes choosing a method for the assessment. In the context of web development it is important to use a method that is easy to learn and intuitive to use. Thereafter, each alternative from the previous step has to be assessed.
- **Assess the lifetime-phase** – The risks and opportunities of the web applications lifetime phase is assessed using the same method as in the previous step.
- **Weight the risks and opportunities and find a suitable action** – Compare the risks and opportunities of both assessments, and discuss how they might interfere with each other. Decide on the action with the most preferred results. This can be done by identifying 3 possible relationships (as shown in figure 2):
  - a positive relationship between the trade-off and the web applications lifetime phase. This is the case when a strength from the trade-off is helping avoiding the threat from the system phase.
  - a neutral relationship between the applications lifetime phase and the trade-off.
  - a negative relationship between the trade-off and the applications lifetime phase. This is the case when a threat from the trade-off can increase the threat from the applications lifetime phase.

5 EXAMPLE

In this section we present an example how an trade-off can be performed. In this example we use a trade-off on the requirement specification practices. Three candidate alternatives for this practise are assess using the the SWOT analysis. This analysis is assessed against a SWOT analysis of the web applications situation. An example of how a trade-off alternative is assessed against the lifetime situation is shown in figure 2. The two SWOT analysis's for the remaining trade-off alternatives are shown in figure 3. For each trade-off alternative a similar trade-off analysis – using a QFD-like matrix – has to be performed.

The stakeholders have to assess the trade-off based on the information in three SWOT analysis's and three trade-off analysis's to decide what is the appropriate trade-off in this situation.

			Appl assessment			
			S	W	O	T
			Can test new functionality	Little or no testing	Web application is easely changed	Users do not return due to poor quality rapidly
Trade-off alternative 1	S	Flexible deployment rutines		−	+	−
		User participation	+			−
	W	Dependent on developers domain knowledge	−		+	
		No verification and validation of requirements		−		
	O	Attracting new users	+			
		Getting early user feedback		+		
	T	Customers are not satisfied		−		

Figure 2: Trade-off evaluation for Trade-off alternative 1.

6 DISCUSSION AND CONCLUSIONS

The motivation for the work presented in this paper is to create an awareness about the trade-offs that are performed during development of web applications and to provide a way of assessing trade-off situations in order to keep the freedom of choice that is associated with trade-offs. In the field of web application development – as it has been described in this paper – this has to be done in a way that brings together all stakeholders, that collects all information that is available, and that let the stakeholders assess it in a way that helps them to make a good decision.

The decision on which method to use when collecting qualitative information from stakeholders is important. The decision is not on the pro's and con's of the method alone, but also on which method is known to the stakeholders and enable them in bringing together their tacit knowledge. In our view, bringing the stakeholders together and let them share their knowledge, beliefs and opinions, is the most important part of our approach. The decision on which method to use for this purpose comes only second to this. We have chosen to use the SWOT analysis, because in our experience it is widely known and easy to use for professionals in developing companies.

Other methods to collect and organise qualitative

SWOT 2, Trade-off analysis 2: Medium TTM and short requirement specification	**SWOT 3, Trade-off analysis 3:** Long TTM and detailed requirement specification
Strenght – better documentation of requirements and therefore more knowledge	**Strenght** – easy communication over requirments, Documented knowledge about user satisfaction
Weakness – no reliable verification and validation of requirements	**Weakness** – harder to get mainstream user involvement (requires user groups), harder to change requirements (it takes time)
Opportunities – can respond to market changes within some time with a better understood solution	**Opportunities** – carefully changing a system and trying to hold on the customer base, mananing user satisfaction
Threats – too long respons when market is changing rapid	**Threats** – slow reaction to changes in market (ex. to a new competitior)

Figure 3: SWOT analysis for trade-off alternatives 2 and 3.

information can be used. Other methods that could have been used are Impact Estimation tables (Gilb, 2005), Affinity diagrams (Straker, 1995) and QFD (Akao, 1990). The advantage of using Impact Estimation tables is that it can be suited to the aspects that are of interest for a project. However, as the information that is collected is qualitative, it is important that not too many aspects are involved. The SWOT analysis uses four aspects, which should be relevant to most projects.

In the future this approach should be validated empirically with respect to its performance – how much qualitative information can be collected and analysed in a reasonable amount of time. Other interesting directions for future research is to study the effect of this approach on knowledge sharing and establishing a common understanding among stakeholders.

In this paper we have shown how trade-off on development practices are performed in web application development, and presented an approach to perform trade-offs on development practices with qualitative information. The objectives behind this approach is to create an awareness for trade-off situations that otherwise will go unnoticed, thereby leaving out opportunities to reach the objectives of a web application.

REFERENCES

Akao, Y. (1990). *Quality Function Deployment*. Productivity Press.

Biffl, S., Aurum, A., Boehm, B., Erdogmus, H., and Grünbacher, P., editors (2006). *Value-Based Software Engineering*. Springer-Verlag Berlin Heidelberg.

Cohen, M. D., March, J. G., and Olsen, J. P. (1972). A garbage can model of organizational choice. *Administrative Science Quarterly*, 17(1):1–15.

Gilb, T. (2005). *Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage*. Butterworth-Heinemann Ltd.

Hanseth, O. (2004). Knowledge as infrastructure. In Avgerou, C., Ciborra, C., and Land, F., editors, *The Social Study of Information and Communication Technology*, pages 103–118. Oxford University Press.

Hill, T. and Westbrook, R. (1997). Swot analysis: it's time for a product recall. *Long Range Planning*, 30(1):46–52.

Lago, P. and van Vliet, H. (2005). Explicit assumptions enrich architectural models. In *Proceedings of the 27th international conference on Software engineering*, pages 206 – 214.

N. Yusop, D. Z. and Lowe, D. (2006). The impacts of non-functional requirements in web system projects. In *Proceeding of European and Mediterranean Conference on Information Systems*.

Nonaka, I. and Takeuchi, H. (1995). *The Knowledge Creating Company*. Oxford University Press.

Ramesh, B., Pries-Heje, J., and Baskerville, R. (2002). Internet software engineering: A different class of processes. *Annals of Software Engineering*, 14:196–195.

Shull, F., Mendonca, M. G., Basili, V., Carver, J., Maldonado, J. C., Fabbri, S., Travassos, G. H., and Ferreira, M. C. (2004). Knowledge-sharing issues in experimental software engineering. *Empirical Software Engineering*, 9(1-2):111–137.

Straker, D. (1995). *A Toolbook for Quality Improvement and Problem Solving*. Prentice Hall.

Ziemer, S., Sampaio, P., and Stålhane, T. (2006). A decision modelling approach for analysing requirements configuration trade-offs in time-constrained web application development. In *Proceedings of SEKE 2006*.

Ziemer, S. and Stålhane, T. (2006). Web application development and quality - observations from interviews with companies in norway. In *Proceedings of Webist 2006*.

Ziemer, S., Stålhane, T., and Sveen, M. (2005). Trade-off analysis in web development. In *3-WoSQ: Proceedings of the third workshop on Software quality*, pages 70–75. ACM Press.