Improvement Strategies for Agile Processes: A SWOT Analysis Approach

Hamed Yaghoubi Shahir, Shervin Daneshpajouh, and Raman Ramsin
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
yaghoubi@ieee.org, daneshpajouh@ce.sharif.edu, ramsin@sharif.edu

Abstract

Agile software development methodologies have been greeted with enthusiasm by many software developers, yet their widespread adoption has also resulted in closer examination of their strengths and weaknesses. While analyses and evaluations abound, the need still remains for an objective and systematic appraisal of Agile processes specifically aimed at defining strategies for their improvement. We provide a review of the strengths and weaknesses identified in Agile processes, based on which a Strengths-Weaknesses-Opportunities-Threats (SWOT) analysis of the processes is performed. We suggest this type of analysis as a useful tool for highlighting and addressing the problem issues in Agile processes, since the results can be used as improvement strategies.

1. Introduction

The widespread adoption of Agile software development methodologies has prompted closer scrutiny of the reasons behind their success [15]. It can now be safely observed that *speed* has been the key factor contributing to the success of Agile methodologies, as it makes them extremely appealing to many companies and organizations which constantly strive to develop software in the shortest time possible. This requirement cannot be readily satisfied by traditional methodologies such as SA/SD and OMT, or modern heavyweights such as RUP and FOOM. Agile methodologies claim to have found a solution to this problem by shrinking the software development process and basing the endeavor on a set of sound principles and practices that speed up the development process. As a consequence, these methodologies gained extensive popularity shortly after their advent, and were widely accepted in the software development community. However, the success has come at a cost, as some fairly standard software development philosophies and practices were sacrificed in the euphoria, some even dismissed as myths. The misunderstandings and ambiguities, the radical leanings of some Agile advocates, and the extreme criticisms expressed by cynics led to the formal announcement of the *Agile Manifesto* and the widely accepted *Agile Principles* [7].

Apart from numerous fiery – albeit enlightening – debates (such as that seen in [6]), Agile methodologies have been extensively and objectively analyzed over the years. Most of these analyses focus on the limitations and weaknesses of the processes [9,11,16,20,21]. However, there have also been numerous research efforts directed towards providing a more balanced appraisal of the methodologies through paying proper attention to their strengths as well [2,10]. What seems to be lacking, however, is a comprehensive review of the capabilities and the limitations that can then be used as a basis for identifying areas for improvement, as well as strategies for attaining the desired results. In this paper, we strive to provide a thorough examination of the known strengths and weaknesses of Agile methodologies, based on the literature and our own observations from studying seven prominent Agile methodologies: DSDM, Scrum, XP, dX, ASD, Crystal Clear, and FDD [19]. The strengths/weaknesses thus identified are then fed into a *S*trengths-*W*eaknesses-*O*pportunities-*T*hreats (*SWOT*) analysis process, which identifies factors external to the methodologies (*opportunities* and *threats*) that can help or hinder their advancement, and ultimately results in improvement strategies.

The rest of this paper is organized as follows: A brief overview of SWOT analysis is presented in Section 2; Section 3 contains a review of the strengths and weaknesses of Agile processes, and section 4 lists the set of opportunities – that can help improve Agile processes, as well as the threats – that can hinder their advancement; Section 5 contains the results of the SWOT analysis and provides strategies for improving Agile methodologies; Conclusions and areas for furthering this research are discussed in Section 6.

2. SWOT Analysis: An Introduction

*S*trengths - *W*eaknesses - *O*pportunities - *T*hreats (*SWOT*) analysis is a strategic planning tool used for analyzing and evaluating the *strength*-, *weakness*-, *opportunity*-, and *threat* points that projects and business ventures face, and thereby helps identify possible strategies for achieving predefined objectives [4]. In identifying these critical aspects, one should note that strengths and weaknesses are generally internal to the business, while opportunities and threats are external factors. Once these four categories of factors are identified, they are placed in a matrix called SWOT, based on which strategies for achieving the objectives are defined.

Figure 1 shows an example SWOT matrix. Four types of strategies can be defined: Strengths-Opportunities (SO), Weaknesses-Opportunities (WO), Strengths-Threats (ST) and Weaknesses-Threats (WT). As the names imply, strategies are formed through combining different factors. For instance, one or more strengths pointed along one or more opportunities may be used to define an SO strategy; in this type of strategy, strength points are utilized in order to make use of the opportunities. In a WO strategy, the purpose is to diminish the weak points by the help of opportunities. In a ST strategy, the strength points are used to reduce the threat points, and finally in a WT strategy, defensive approaches are used to cover the weaknesses and to avoid threats.

Even though mainly used in marketing and strategic business planning, SWOT analysis is quite useful in other applications as well, especially for research aimed at improving processes: it helps concentrate on objectives and strengths/weaknesses, identify potentialities for improvement, and define strategies for achieving the desired outcome [4]. It should be noted, however, that the main focus will inevitably be on WO strategies, since they are the most relevant to process improvement.

3. Strengths and Weaknesses of Agile Methodologies

A review of the properties of Agile methodologies is presented in this section, according to which the strengths and weaknesses of these methodologies will be analyzed. Several research efforts have been conducted on Agile methodologies and have established their strengths and weaknesses [13,21]. In addition, the Agile Manifesto and the Agile Principles [7] can be used as useful pointers. The strengths and weaknesses thence compiled are presented in the following subsections.

3.1. Strengths

Strengths of Agile methodologies have been stressed time and again, yet there is still need for a recap of what has been observed in this regard:

1- **Welcoming Changing Requirements, Even Late in Development:** As amply stated in [11]: "[Agile methodologies see] change as an ally rather than an enemy. Change allows for more creativity and quicker value to the customer." Highsmith states in [14] that Agile methodologies are suitable for projects that demonstrate high variability in tasks.

2- **Satisfying Stakeholders and Users:** The presence of stakeholders in the development team causes their viewpoints to be taken into account when development decisions are made, typically reducing the need for later rework. On the other hand, by valuing the users' viewpoints (even if not implementing them all), they consider themselves a part of the development team and tend to fully commit themselves to advancing the project.

3- **Iterative-Incremental Development:** Iterative-incremental development is a characteristic of all Agile methodologies: To achieve frequent delivery, an iterative-incremental development engine with short iterations seems like the logical choice. Yet it also provides benefits of a rather more subtle nature which are valuable per se; enhanced risk management and quality assurance are notable examples.

4- **Simplicity:** Striving for simplicity is an important Agile principle. Simplicity can be considered both as an advantage and a disadvantage for this group of methodologies. In fact, in small projects, design simplicity is an advantage since it reduces the overheads and the time the project takes, whereas it can prove a disadvantage in larger projects, where more rigorous processes are required. There are three different aspects related to simplicity in Agile methodologies:

   a) **Process Simplicity:** Agile methodologies incorporate lightweight processes.

   b) **Design Simplicity:** Agile design is mostly limited to informal architectural design and elaborate (albeit ad hoc) program design.

   c) **Code Simplicity:** Simplifying the code enhances its intelligibility, and is therefore instrumental in guaranteeing the level of flexibility and maintainability needed in Agile development. Moreover, since there is little stress on documentation in Agile processes, the code itself becomes a major communication medium.

5- **Test-Driven Development:** Most Agile methodologies focus on coding and testing rather than analysis and design, to the extent that in many Agile methodologies, test cases and test code are generated before the coding commences on the actual release. The main advantages of test-based coding are: improved code maintainability, reduction in inconsistencies, and enhanced quality assurance [16].
6- **Pair Programming:** In some Agile methodologies, pair programming has been considered an enabling factor for Agile development, and has been explicitly or implicitly fused into the process. Pair programming means that developers work in pairs, performing all their tasks together. As stated in [5] and [11], these tasks are not limited to programming, as the two programmers continually collaborate on the same design, algorithm, code, or test as well. This way, there are two people responsible for a task and they can cover each other's weaknesses. Nevertheless, although pair programming is considered as a strength of Agile methodologies, it can also cause quite a few problems, as mentioned in [1].
7- **Refactoring:** Because of some of the defining characteristics of Agile methodologies, such as their concentration on the code rather than on the design, the need for dynamicity and circulation of the team members, and the requirement for the product to be highly flexible, refactoring is essential and inevitable in Agile processes. In this context, the main goals of refactoring are: removing redundant and unnecessary code, increasing code simplicity, achieving flexibility without any change to the behavior of the system, and improving communication among developers.
8- **Frequent Integration:** In most of the Agile methodologies studied, integration occurs continuously during the development and production processes. This is because the development process in Agile methodologies is iterative-incremental, and executable increments are released in the very first iteration. To achieve this, and also to be able to test the product in the user environment and receive feedback, integration becomes an essential activity.
9- **Dynamicity of the Development Team:** Team members involved in the project are constantly reallocated and interchanged, with the following advantages:
   a) Dynamicity will result in a better flow of information among team members.
   b) Dynamicity reduces the dependency of the project on a limited number of team members.
   c) The team's productivity is increased considerably [9].
10- **Effective Planning:** Since most Agile methodologies are iterative-incremental, planning is an issue taken very seriously. Agile methodologies are also extremely wary of the "Death by Planning" risk encountered in non-Agile processes, and therefore stress the need for regular plan reviews.
11- **Reflection and Retrospective Review:** Reviewing the completed tasks and the deliverables mainly aims at making assessments as to how the project is progressing in order to obtain accurate estimations for the next iterations of the development process [11]. It also helps verify that the project is on the right track, and may even focus on verifying the efficiency of the development process itself.
12- **Prioritizing Requirements:** Prioritizing the requirements (according to the risks associated and/or their value to the customer) is a common practice in Agile methodologies, mainly because it facilitates frequent release of executable software, provides better support for iterative-incremental production of software, helps mitigate project risks, and helps focus on satisfying the stakeholders.
13- **Teamwork and Collaborative Decision Making:** Collaborative decision making means that the opinion of each team-member can affect the final decision. In other words, the power of decision making is distributed among the project managers and developers [16]. Nevertheless, in such circumstances, project managers should be more careful of the "Design by Committee" and "The Grand Old Duke of York" syndromes, and also make sure that the project does not deviate from its main path.
14- **Rapid Development:** In Agile methodologies, we intend to increase the speed of the development by overlooking some unnecessarily rigorous tasks. Naturally, these methodologies are not appropriate for every project, and are certainly not expected to be. What is considered a strength here is the ability of Agile methodologies to do what they do *fast*.

3.2. Weaknesses

By weaknesses, we mean areas where Agile methodologies have been shown to need improvement. The weaknesses are listed below, with a more in-depth account of the particulars of each provided:

1- **Inefficiency of Interaction and Communication Methods:** The prevalent type of interaction in Agile methodologies is face-to-face communication. Although other kinds of communication media (such as sticky notes and whiteboards) are also used to exchange ideas, lack of models and documented design leads to insufficient references in case disagreements occur or a state of oblivion develops. Furthermore, human interaction is prone to various anomalies by its very nature, and therefore not adequate as the main communication medium in software development efforts.

2- **Limitations in Global or Distributed Developments:** In Agile methodologies, development-team members work in proximity to each other; some processes even require that the whole team be collocated, i.e. in the same room or building. This is due to the fact that face-to-face communication, daily and weekly meetings, and human interactions have a critical role in the success of such processes. Although Agile developers stress the applicability of Agile processes to distributed applications, the evidence seems to point to these applications as a likely problem area for Agile methodologies [9,11,16,20].
3- **Need for Customer Presence during Development:** Developing and producing software is the responsibility of the development team. Forcing user participation may not be acceptable or even possible in many organizations.
4- **Heavy Reliance on Development Team:** As stressed in the Agile Manifesto, Agile methodologies are more people-oriented rather than process-oriented. Consequently, situations develop where teamwork issues become more problematic that they normally should [9,11].
5- **Lack of Documentation and Modeling:** Agile methodologists believe that project knowledge should be in the participants' heads rather than on paper. This causes every item that is considered non-essential to be omitted. These may even include essential analysis and design documents. It should be noted, though, that complete omission is not possible, and few projects (mostly simple and small projects) can be developed this way. The extreme views enforced by some Agile methodologies, however, impose unnecessary restrictions on the applicability of these processes.
6- **Products Suffering from Deficiency in Reusability:** As pointed out in [21], Agile processes are mainly targeted at developing custom software. Developing generalized solutions or products facilitating future development projects is usually sacrificed in order to gain higher development speed, and reusability suffers as a consequence. Agile principles emphasize early and frequent delivery of working software, rather than developing software made up of reusable or general components. In fact, lack of design and modeling restrains reusability and generality: It is in the design and modeling phases that generalization and reusability can be provisioned for and achieved. Furthermore, producing reusable components needs unambiguous and precise "Quality Control", which is typically not supported by Agile methodologies. This type of quality control is necessary to prevent the propagation of errors.
7- **Misestimation of Project Time and Budget:** Allowing frequent changes to the requirements is an Agile principle, yet it complicates the estimation of project time and cost [13]. In addition, due to the lack of modeling and design processes, estimating the workload is difficult. For example, the absence of analysis class diagrams usually means that the number of the classes to be implemented remains unknown until downstream phases. Therefore, project managers cannot perform adequate planning, and project plans have to be changed frequently.
8- **Limitations in Subcontracting and Outsourcing:** Lack of precise documented requirement specifications causes difficulty in using Agile methodologies for outsourcing and subcontracting [21]. Outsourcing and subcontracting need precise contracts, while in Agile processes requirements are allowed to change frequently, even late in the course of development. In addition, some organizations do preliminary design themselves and order a product afterwards; lack of design reduces the applicability of Agile methodologies in such situations.
9- **Limitations in Developing Safety-Critical Software:** Agile methodologies alone are not sufficient for developing safety critical software, as quality control mechanisms incorporated in Agile methodologies do not provide the (mostly formal) features required for this purpose [5,12,21]. Some Agile features, such as the test-driven approach and the early delivery of working software, are useful practices in this regard, but they are by no means sufficient.
10- **Limitations in Developing Large and Complex Software:** Refactoring is a very useful technique in software development. However, Agile methodologies assume that the need for design can be replaced by refactoring [21]. Although such an assumption is possible for small to medium-sized software, it is not suitable for large and complex systems, where a central architecture and detailed design models are essential.
11- **Limitations in Managing Large Teams:** Agile methodologies are able to manage, control, and coordinate small to medium-sized teams. Agile communication mechanisms are also suitable for

such team sizes. As the team size increases, Agile mechanisms fail to act effectively. For example, informal face-to-face communication and management and holding stand-up meetings and review sessions are not readily possible in large teams [21].

12- **Lack of Metrics and Measures:** The metrics typically encountered when applying Agile processes are the Project Velocity, and the ratio of implemented features (such as requirements, user stories, features, etc.) to elicited features. Obviously, these metrics are not sufficient for measuring a project's progress.

13- **Heavy Dependence on Tools:** One of the problems in using Agile methodologies is their heavy dependence on tools. XP, for instance, is completely based on "Collective Code Ownership", and therefore is not applicable and successful without the appropriate support tools. Practices such as test-driven development are essentially tool-dependent, and since Agile development processes are dependent and based on such techniques, almost no Agile methodology is practical without such tools.

14- **Insufficient Guidelines for Testing:** As we saw in the previous section, Test-Driven Development (TDD) is an Agile strength; however, it can also be problematic at times, mainly due to lack of adequate guidelines [5,9].

4. Opportunities and Threats

In this section, a number of opportunities and threats relevant to Agile methodologies are listed. It should be mentioned that not all such opportunities and threats have been identified. We have focused on opportunities which can improve the weaknesses of Agile methodologies, and similarly threats which can be reduced using the opportunities.

4.1. Opportunities

The opportunities identified are as listed below:

1- **Methodology Engineering:** It has been observed that a single methodology is not suitable for all situations. Hence, *Methodology Engineering* – or *Method Engineering (ME)*, as it has come to be called – has been proposed as a way to develop, adapt, configure, or enhance methodologies [18]. Of the three most common ME approaches: *Assembly-based* (assembling method components retrieved from method repositories), *Paradigm-based* (instantiating a process meta-model), and *Extension-based* (enhancing an existing methodology with new features), the latter provides suitable opportunities for targeting the weaknesses of existing methodologies. The following extension means (or extension *patterns*) can be identified as useful tools in this context:

a) *Agile Modeling (AM)* and *Agile Model-Driven Development (MDD)*: A collection of practices and principles for adding simple modeling activities to Agile methodologies [3].

b) *Feature Driven Development* augments: Feature Driven Development (FDD) is an Agile methodology core, later augmented with *Project Management*, *Configuration Management*, and *Quality Assurance* features [17]. These augments can also be used for enhancing other Agile methodologies.

c) *Scrum* extension patterns: The *Scrum* methodology relies on efficient inter-team and intra-team communication. It therefore requires special provisions when multiple and distributed teams are involved. The original Scrum methodology has been augmented with various *Project Management*, *Complexity Management*, *Communication Management*, *Planning*, and *Scheduling* features. These extensions can be applied to other Agile methodologies as well.

2- **Light Analysis and Design:** In a *light* analysis and design approach, only the most basic and essential models (such as those depicting the use cases and classes) are utilized during analysis and design. Some Agile processes – such as ASD and FDD – already include such practices, thereby remedying the chronic model-phobia with which older Agile methodologies were afflicted.

3- **Expert Advice:** Using expert opinion in different contexts is an opportunity that should be put to maximum use. This not only refers to development experts, who are involved extensively in Agile teams, but also includes domain-, technology- and, above all, methodology experts, who can help tune the methodology to fit the project at hand.

4- **Distributed Software Development Strategies and Techniques:** The daily expansion of the Internet and the constant increase in its speed has made distributed software development a modern trend. Teleconferencing technologies and web-based development environments are becoming increasingly popular because of their availability and usability: local installation is not required and the customer can review a sample of the product simply by connecting to the central server used by the development team.

5- **Reverse Engineering:** It is often the case that the implemented classes are different from the ones that were originally designed. This difference can be in terms of class attributes and methods, or even simply as an undesirable increase in the number of the classes. Reverse engineering tools can be used so that the models are developed quickly (in less than a few minutes) based on the code.

6- **Standardized Testing Methods:** Years of experience in developing software has led to compilation and standardization of different testing methods. Such proven techniques can and should complement TDD practices, and can have a profound effect on Q/A support in Agile development.

4.2. Threats

Threats facing Agile processes in the software world are mainly limited to cases where these processes face fierce competition and/or skepticism. We have elaborated on one such important instance:

1- **Lack of Interest in Utilization of Agile Methodologies in Traditional Organizations:** As aptly stated in [11], Agile methodologies have not been received well in traditional organizations, mainly due to difficulties in coordinating traditional and Agile processes and/or human resources, and difficulties in conforming to standards such as CMM.

5. SWOT Analysis

In the previous sections, the strengths, weaknesses, opportunities and threats relevant to Agile methodologies were identified. In this section, we present strategies for improving the processes by putting these factors in a SWOT matrix, as shown in Figure 1. What we are trying to demonstrate is the usefulness of the SWOT analysis approach in addressing the problem issues.

5.1 Objective

The ultimate SWOT analysis objective – although rather ambitious – is to improve agile methods so that they can replace their heavyweight counterparts.

With this objective in mind, and considering the strengths, weaknesses, opportunities, and threats identified, all the suggested improvement strategies are in the W-O category. While the other three categories can be of merit in this context, we have intentionally limited our focus to the category that is most relevant to our ultimate intention, i.e., process improvement. The results are explained in the following subsection.

5.2. W-O Strategies

W1, O1, O2: Using light analysis/design and ME to ameliorate the weaknesses in information interchange. Since light analysis/design stays loyal to the Agile manifesto, it can be used as a solution to information exchange problems in Agile methodologies. Incorporating analysis/design activities reduces misunderstandings and personal misinterpretations. Method engineering can also be used to extend and adapt an existing methodology to improve communications; Scrum extensions proposed in [8] can be applied to this aim. Agile Modeling (AM) can also improve communications by providing models to be used as information interchange media.

W2, O1, O2, O3, O4: Using light analysis and design, expert advice, distributed software development strategies and techniques, and ME extension patterns (Agile Modeling and Agile Model-Driven Development) to overcome the problems caused by distributed and global development of the product.

W3, O4: Using distributed software development strategies and techniques to help with the problem of the customer not being present at the development team's location, and vice versa.

W4, O1, O2: Using light analysis and design, and Agile Modeling to reduce the weaknesses caused by over-reliance on the development team. Analysis and Design helps improve organization and distribution of the tasks, and also gives the development team members a better knowledge of their responsibilities.

W5, O1, O2: Using light analysis and design, and Agile Modeling extensions to overcome the lack of modeling and documentation. Light analysis and design can be used for providing the minimum modeling and documentation required.

W6, O1, O2, O5: Using light analysis/design, reverse engineering tools, and Agile Modeling extensions for producing a reusable product. A basic principle for creating a reusable product is to produce a blueprint for the structure (*architecture*) and the classes of the system. This can be done using light analysis and design at the early stages of the software development process. Models should remain consistent during later stages, yet implementation teams (being agile) tend to overlook this requirement. Reverse engineering tools can be used for producing/updating the models.

W7, O1, O2: Using light analysis and design, and extension-based ME (through applying FDD project-management augments) for estimating the amount of work required, resulting in better scheduling and time estimation in each iteration. It is clear that having a design gives a better understanding of the amount of work required, giving the project manager a better knowledge of the work, hence helping him in obtaining a better estimation of the implementation time and the cost of the project.

W8, O1, O2: Using light analysis/design, Agile Modeling, Agile MDD, and communication-management extensions to enable outsourcing. This means that a better understanding of the product is achieved in the early stages of its production, thereby facilitating the outsourcing of the whole product or parts of it.

W12, O2: Using light analysis and design to overcome the weakness in measurements and using metrics.

W14, O6: Using existing standardized testing methods to overcome the weakness caused by lack of sufficient guidance on testing methods in Agile methodologies.

5.3. Summary of SWOT Analysis

Some of the strengths and weaknesses of Agile methodologies were compiled and discussed; furthermore, the opportunities and threats related to these methodologies were identified. By feeding these into a SWOT analysis process, we have strived to demonstrate how improvement strategies can be obtained.

Ten improvement strategies were presented for fourteen weaknesses. It seems that the remaining weaknesses cannot be resolved based on the opportunities identified herein, as some basic Agile principles are contradicted when applying the opportunities to overcome these weaknesses; agility is therefore jeopardized.

Through this research, we have come to the same conclusion as that reported by Boehm in [6]: Agile and Traditional approaches complement each other, and convergence attempts are therefore beneficial to both parties. The best solution to the Agile-Traditional confrontation seems to be to find a balance between Agile and Traditional features. Indeed, *light analysis and design* and *methodology engineering* approaches are suggested as solutions in most of our improvement strategies; this brings Agile processes closer to Traditional processes. In other words, the analysis applied herein seems to confirm that striving for a balance between Agile and Traditional methodologies is likely to be feasible, and worthwhile.

	Strengths S1. Welcoming changing requirements even late in development S2. Satisfying stakeholders and users S3. Iterative-incremental development S4. Simplicity S5. Test driven development S6. Pair programming S7. Refactoring S8. Frequent integration S9. Dynamicity of the development team S10. Effective planning S11. Reflection and retrospective review S12. Prioritizing requirements S13. Teamwork and collaborative decision making S14. Rapid development	**Weaknesses** W1. Inefficiency of interaction and communication methods W2. Limitations in global or distributed developments W3. Need for customer presence during development W4. Heavy reliance on development team W5. Lack of documentation and modeling W6. Products suffering from deficiency in reusability W7. Misestimation of project time and budget W8. Limitations in subcontracting and outsourcing W9. Limitations in developing safety critical software W10. Limitations in developing large and complex software W11. Limitations in managing large teams W12. Lack of metrics and measures W13. Heavy dependence on tools W14. Insufficient guidelines for testing
Opportunities O1. Methodology engineering O2. Light analysis and design O3. Expert advice O4. Distributed software development strategies and techniques O5. Reverse engineering O6. Standardized testing methods	**S-O Strategies** Out of the Scope of this Research	**W-O Strategies** W1,O1,O2 W2,O1,O2,O3,O4 W3,O4 W4,O1,O2 W5,O1,O2 W6,O1,O2,O5 W7,O1,O2 W8,O1,O2 W12,O2 W14,O6
Threats T1. Lack of interest in utilization of Agile methodologies in traditional organizations	**S-T Strategies** Out of the Scope of this Research	**W-T Strategies** Out of the Scope of this Research

Figure 1. SWOT Matrix and Proposed W-O Strategies

6. Conclusion

Our main objective has been to highlight the usefulness of SWOT analysis as a process improvement tool, specifically targeting existing Agile methodologies. In striving to achieve this objective, we have limited our focus to one category of SWOT strategies (namely, addressing Weaknesses through utilizing the Opportunities), since it is the most relevant to the task at hand. We have thereby suggested concrete strategies for improving Agile processes. Many of the results may not seem novel or revolutionary, yet the analysis approach is shown to be a promising means for process improvement.

This research can be furthered by attempting to implement the strategies in existing Agile methodologies, and thereby producing concrete Agile methodologies with improved features. Perfecting the SWOT matrix should be an ongoing process aiming to stay current with new trends, be they considered opportunities or threats.

7. Acknowledgements

We wish to thank the Research Vice-Presidency of Sharif University of Technology for sponsoring this research.

8. References

[1] Abrahamsson, P., "Pealing the Hype into Pieces: What Do We Really Know about Agile in Research and Practice?", VTT Technical Research Centre of Finland, Oulu, Finland, Available at: http://www.agile-itea.org/public/papers/OLIO_abrahamsson.pdf, 2006.

[2] Abrahamsson, P., Warsta, J., Siponen, M.T., and Ronkainen, J., "New Directions on Agile Methods: A Comparative Analysis". In *Proceedings of 25th Conference on Software Engineering (ICSE)*, 2003, pp. 244-254.

[3] Ambler, S. W., Agile Modeling, Available at: *http://www.agilemodeling.com*, visited in December 2007.

[4] Armstrong, M., *Management Processes and Functions*, CIPD, London, 1996.

[5] Beck, K., and Andres, S., *Extreme Programming Explained: Embrace Change*, 2nd Edition, Addison-Wesley, Reading, MA, 2004.

[6] Beck, K., and Boehm B., "Agility through Discipline: A Debate", *Computer*, IEEE, Vol. 36, No. 6, June 2003, pp. 44-46.

[7] Beck, K., et al., Principles behind the Agile Manifesto, Available at: *http://www.Agilemanifesto.org/principles.html*, visited in May 2007.

[8] Beedle, M., Devos, M., Sharon Y., Schwaber, K., and Sutherland, J., "SCRUM: An extension pattern language for hyperproductive software development", Available at: *http://jeffsutherland.com/scrum/scrum_plop.pdf*.

[9] Berard, E.V., "Misconceptions of the Agile Zealots", The Object Agency, L.L.C., Available at: *http://www.svspin.org/Events/Presentations/MisconceptionsArticle20030827.pdf*, 2003.

[10] Boehm, B., and Turner, R., *Balancing Agility and Discipline: A Guide for the Perplexed*, Addison-Wesley, Reading, MA, 2004.

[11] Boehm, B., and Turner, R., "Management Challenges to Implementing Agile Processes in Traditional Development Organizations", *IEEE Software*, Vol. 22, No. 5, September 2005, pp. 30-39.

[12] Boehm, B., "Some Future Trends and Implications for Systems and Software Engineering Processes", *Systems Engineering*, Vol. 9, No. 1, 2006, pp. 1-19.

[13] Coram, M., and Bohner, S., "The Impact of Agile Methods on Software Project Management", In *Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS)*, April 2005, pp. 363-370.

[14] Highsmith, J., "Agile Project Management: Principles and Tools", Cutter Consortium, February 2003.

[15] Maurer, F., and Melnik, G., "Agile Methods: Moving Towards the Mainstream of the Software Industry", In *Proceedings of 28th International Conference on Software Engineering (ICSE)*, 2006, pp. 1057-1058.

[16] Nerur, S., Mahapatra, R., and Mangalaraj, G., "Challenges of migrating to Agile methodologies", *Communications of the ACM*, Vol. 48, No. 5, May 2005, pp. 72-78.

[17] Palmer, S. R., and Felsing, J. M., *A Practical Guide to Feature-Driven Development*, Prentice-Hall, Englewood Cliffs, NJ, 2002.

[18] Ralyté, J., Deneckére, R., and Rolland, C., "Towards a generic model for situational method engineering". In *Proceedings of CAiSE 2003 (LNCS 2681)*, 2003, pp. 95-110.

[19] Ramsin, R., and Paige, R. F., "Process-Centered Review of Object Oriented Software Development Methodologies", *ACM Computing Surveys*, Vol. 40, No. 1, February 2008, pp. 3:1-89.

[20] Taylor, P.S., Greer, D., Sage, P., Coleman, G., McDaid, K., and Keenan, F., "Do Agile GSD Experience Reports Help the Practitioner?", In *Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner*, 2006, pp. 87-93.

[21] Turk, D., France. R., and Rumpe, B., Assumptions Underlying Agile Software-Development Processes, *Journal of Database Management*, Vol. 16, No. 4, 2005, pp. 62-87.