

SIX ASPECTS OF AN AGILE SOFTWARE DEVELOPMENT METHODOLOGY

Asif Qumer, Brian Henderson-Sellers
University of Technology

Abstract

A shared vision of an agile methodology can play a vital role in adaptive agile software development environments as the absence of a shared or common vision is one of the main factors of software project failures. This paper presents six aspects of an agile software development methodology: agility, abstraction, process, people, product and tools. This set of aspects is an attempt to provide a guiding vision or mental-model for an agile methodology. The purpose of this paper is to explicitly describe in detail these aspects that are part of our agile software solution framework (ASSF). These six aspects can be combined to generate various situation-specific configurations of agile methodologies by using a method engineering approach.

Keywords: *Agile Methods/Methodology*

1 INTRODUCTION

It is well acknowledged that a pre-defined or fixed agile methodology cannot be used off-the-shelf for any specific software project. There are a number of agile methods that have been proposed over the last decade and offer many powerful agile practices that can be tailored or combined by using a standard agile mental-model or meta-model for the engineering of a situation-specific hybrid agile software development process. This paper highlights the need for a standard agile mental-model and provides a standard set of aspects for an agile software development methodology. These related six aspects, as identified here, represent a shared agile methodology guiding vision, which is essential for driving behaviours in agile software development projects (Augustine 2005) – a guiding vision that should be continually refined (Augustine and Woodcock 2003).

Within the overall context of situational method engineering (Kumar and Welke, 1992), this project focusses on providing conceptual and practical support for agility in methods. Within such an overall framework, we are currently developing an agile software solution framework (ASSF). ASSF provides a comprehensive set of agile methodology aspects to create, tailor and customize multi-abstraction or m-oriented (object-oriented, agent-oriented, aspect-oriented, service-oriented etc.) agile processes.

ASSF is also a first step towards the future development of a meta-model for agile software development methodologies. There are a number of meta-models (OPF, ISO, and Australian Standards) for traditional approaches to software development but they do not explicitly discuss the new aspect of agility, which is an essential aspect for an agile software development method. According to our survey and focus group, it is also found that existing standards do not explicitly discuss the aspect of abstraction either - a critical aspect of any software development methodology (Qumer and Henderson-Sellers 2006b).

This paper presents full details of the core part of our new version of ASSF that provides support for multi-abstraction project development. This paper is organized as

follows: Section 2 provides an overview of the agile software solution framework (ASSF V.2.0). Section 3 describes the research methodology. Section 4 details the six aspects of an agile software development methodology (summarized in Qumer and Henderson-Sellers, 2007a). Finally, Section 5 presents conclusions and future research objectives.

2 THE AGILE SOFTWARE SOLUTION FRAMEWORK (ASSF)

ASSF has four main parts: agile cells, agile toolkit, business value and agile alignment bridge (business-agile), and business. This paper focuses on the first of these four: the core agile cells that represent the six different, but related aspects of an agile method/methodology (Figure 1). The “*” mark (in the last compartment of each cell) shows that anything can be added or modified (whenever required), as required and can be used and localized according to the needs of the local environment. Each aspect of an agile methodology is described in terms of three components: concepts, to specify the relevant emerging concepts; guiding vision mental-models, representing emerging models based on concepts; and enactment, which specifies the people (with responsibilities and accountabilities) who will construct, govern, manage and use these models (“shared guiding vision”) for a specific situation or project.

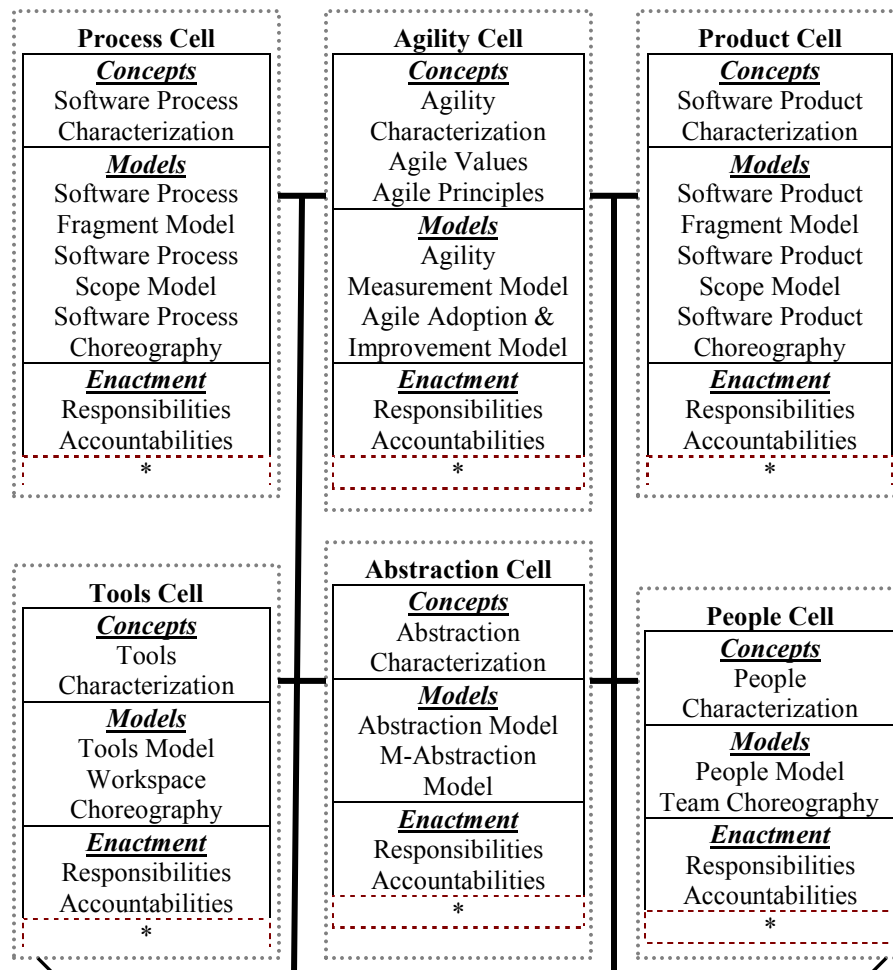


Figure 1. Guiding Vision Model - Six Aspects of an Agile Methodology

3 Research Methodology

The set of agile aspects has been elucidated by the application of an iterative, inductive and interactional mechanism of data collection and instantaneous analysis and emergent interpretation by using a grounded theory research methodology (Glaser and Strauss 1967, Strauss and Corbin 1990, Pandit 1996). We have used and analysed the feedback from a focus group and from individual interviews, our own experience, a number of appropriate extant theories, software development frameworks, methods, models such as AgileManifesto (2006), Agile Project Management (Augustine and Woodcock 2003, Augustine 2005), Extreme Programming (Beck 2000), Feature Driven Development (Palmer and Felsing 2002), Adaptive Software Development (Highsmith 2000), Dynamic Software Development Method (DSDM 2003), Scrum (Schwaber and Beedle 2002), Tropos (Giunchiglia, Mylopoulos, and Perini 2002), Gaia (Wooldridge, Jennings and Kinny 2000), ROADMAP (Juan, Pearce and Sterling 2002), MaSE (Wood and DeLoach 2000), Prometheus (Padgham and Winikoff 2004), PASSI (Cossentino 2005), Service-

Oriented Modelling and Architecture (Arsanjani 2004), Agent Service-Oriented Architectural Design (Cao, Zhang and Ni 2005), SPICE (2003), CMMI (2006), ISO/IEC (2007), OPF (Firesmith and Henderson-Sellers 2002) and Australian Standards (2004) that have relevance to the emerging, data-grounded conceptual categories of software development methodologies to identify the six aspects of an agile software development methodology: agility, abstraction, people, process, product and tools. We found that the first of these aspects (agility) of a software development methodology has not been considered before (Qumer and Henderson-Sellers 2006b) and that the second (abstraction), whilst the keystone of modelling, may need to be considered more explicitly once multi-abstractions are considered.

4 Six Aspects of an Agile Software Development Methodology

This section expounds on the full details of the six identified aspects of agile software development methodologies as depicted in Figure 1: abstraction, agility, people, process, product and tools.

4.1 Abstraction

Abstraction is a critical aspect of any software development methodology that represents a logical view of real world problems or entities or components in a software system such as process-oriented, data-oriented, object-oriented, agent-oriented, aspect-oriented, service-oriented etc. Abstraction includes two shared guiding vision models: abstraction model and m-abstraction model (Figure 2). Responsibilities and accountabilities (people or teams) are associated with these models at the time of model-enactment and instantiation for a specific situation or project. The “*” mark (in the last box) shows that any new abstraction mechanism(s) can be added or modified, as required.

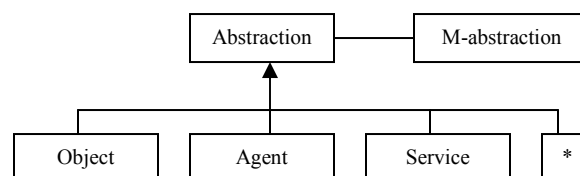


Figure 2. Guiding Vision Model - Abstraction

4.1.1 Abstraction Model

The abstraction model provides a shared guiding vision to agile teams for the development of a single abstraction project (object or agent or service-oriented development). This abstraction model specifies the characteristics and behaviours of a specific abstraction within the context of a software development.

4.1.2 *M-Abstraction Model*

The m-abstraction model provides a guiding vision for a specific situation or project that *combines* more than one abstraction mechanisms to form a hybrid abstraction. It represents the interactions of concerned abstractions for a specific project or situation.

4.2 Agility

The agility aspect includes the concept of agility, agile values and principles and includes two models: an agility measurement model (AMM) (Figure 3), and an agile adoption and improvement model (AAIM) (AgileManifesto 2006, Qumer and Henderson-Sellers 2006a). Responsibilities and accountabilities (people or teams) are associated with these models at the time of model-enactment and instantiation for a specific situation or project.

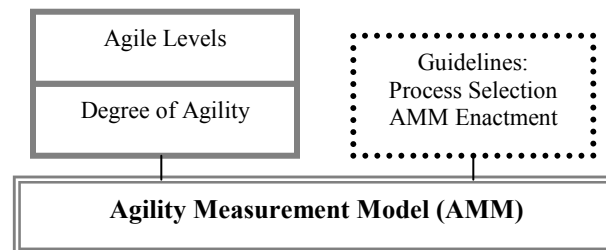


Figure 3 Guiding Vision Model – Agility Measurement Model

4.2.1 *Agility Measurement Model*

The agility measurement model (AMM) facilitates agile teams in the measurement of the degree of agility and in the selection of an appropriate agile level for an agile method for a specific project, since too little or too much agility may be harmful, irrelevant or just inappropriate and unhelpful.

4.2.2 *Agile Adoption and Improvement Model*

The agile adoption and improvement model (AAIM), along with the AMM, will guide an organization or a software development team to effectively (step-by-step) adopt and improve agile practices within a specific organization for a specific situation or project.

4.3 People

The people vision characterizes the attributes of agile people and agile teams (Table 1). It provides two models: an agile people model and agile team choreography model. Responsibilities and accountabilities (people or teams) are associated with these models at the time of model-enactment and instantiation for a specific situation or project.

Guiding Vision: Agile People and Team
<ol style="list-style-type: none"> 1. Education: The special agile software development related education 2. Technical Skills: The demonstrable technical skills (related to agile development) 3. Experience: The evidence of prior experience. 4. Communication and collaboration skills: The ability to work in a communications-oriented environment. 5. Problem solving and self organizing skills: The ability to take own initiatives. 6. Degree of agility: The quantitatively measured value of the degree of agility by using an AMM 7. Business value: The possible business value he/she will deliver 8. Alerts: Any special attached circumstances or behaviours of an agile person.

Table 1. Agile People

4.3.1 People Model

The people model provides a set of criteria outlining the specific attributes of agile people (Table 1), which facilitates self-organized and empowered agile teams to make decisions about the inclusion or exclusion of any specific agile team member in an agile team.

4.3.2 Team Choreography Model

The agile team choreography model provides a mental-model to specify the composition of an agile team in an agile software development organization for a particular situation or project. We have developed an agile team mental-model, EPP (“enhanced pair programming”), and tested this on our pilot projects in industry (Qumer and Henderson-Sellers, 2007b). In EPP, there are a minimum of two developers (one senior and one junior developer) with an option to add another extra developer, if required. Each pair is led by a senior developer and there could be more than one pair in any one agile team which is led by a team lead. In this structure, the agile manager works as a leader and facilitator to empowered agile team members.

4.4 Process

The process aspect includes three models of a software process: a software process fragment model, a software process scope model and software process/method choreography. The process aspect represents various types of processes such as development process, project and process management process, risk management process etc. Responsibilities and accountabilities (people or teams) are associated with these models at the time of model-enactment and instantiation for a specific situation or project.

Guiding Vision: Agile Process Fragment
<ol style="list-style-type: none"> 1. ID & Name: The unique ID and name of the process fragment. 2. Description & Purpose: The related details and purpose of the process fragment. 3. Abstraction: The support for a particular abstraction mechanism (object, agent, service etc.). 4. Tools & People: The requirement of the tools and people to successfully use the process fragment. 5. Development Style: The requirement of a style (iterative, rapid etc.). 6. Physical Environment: The requirement of the physical environment (co-located or distributed). 7. Pre and Post Conditions: The pre and post conditions that must be true. 8. Constraints and Risks: The possible associated constraints and risks. 9. Degree of agility: The degree of agility (by using AMM) of the process fragment. 10. Business Value: The business value added by the process fragment. 11. Alerts: Any special situations describing when the process fragment should <i>not</i> be applied.

Table 2. Agile Process Fragment Model

4.4.1 Software Process Fragment Model

The software process fragment model provides a set of attributes for an agile process fragment (Table 2) for the purpose of guiding self-organizing agile teams to adopt or discard any specific agile process fragment for a particular situation or project.

4.4.2 Software Process Scope Model

The software process scope model represents the scope of an agile process/method at a higher level and can be used by self-organizing agile teams (along with the software process fragment model) for the selection of an agile process for a particular situation or project. The main attributes in this model are: process agility, project size, preferred team size, business culture, code style and technology environment.

4.4.3 Software Process Choreography Model

Software process is collection of software process fragments and a choreography model provides a mental-model or meta-model to specify the composition of an agile process for a particular situation or project. The software process choreography model guides and aligns agile team behaviour in an agile development environment.

4.5 Product

The product aspect has three models of a software product: a software product fragment

model, a software product scope model and software product choreography. The product aspect represents various types of software applications such as object-oriented, agent-oriented, service-oriented etc. Responsibilities and accountabilities (people or teams) are associated with these models at the time of model-enactment and instantiation for a specific situation or project.

Guiding Vision: Software Product Fragment
<ol style="list-style-type: none"> 1. ID & Name: The unique ID and name of the product fragment. 2. Description & Purpose: The related details and purpose of the product fragment. 3. Abstraction: The particular abstraction mechanism (object, agent, service etc.). 4. Tools & People: The requirement of the tools and people to successfully use/develop the product fragment. 5. Development Style: The requirement of a style (iterative, rapid etc.). 6. Physical Environment: The requirement of the physical environment (co-located or distributed). 7. Pre and Post Conditions: The pre and post conditions that must be true. 8. Constraints and Risks: The possible associated constraints and risks. 10. Business Value: The business value that is added by the product fragment. 11. Alerts: Any special situations describing when the product fragment should <i>not</i> be used/ developed.

Table 3. Software Product Fragment Model

4.5.1 Software Product Fragment Model

The software product fragment model represents the characteristics of a software product component (Table 3) for the purpose of guiding self-organizing agile teams to use or develop any specific software product fragment for a particular situation or project.

4.5.2 Software Product Scope Model

The software product scope model describes the characteristics of a software product at a higher level for guiding self-organizing agile teams (along with the software product fragment model) in a particular situation or project. The main attributes in this model are: product size, complexity, criticality, development time and the technology environment.

4.5.3 Software Product Choreography Model

Software product is collection of software product fragments and a product choreography model that provides a mental-model or meta-model to specify the composition (integrated software components or a high-level product architecture and configuration) of a software product for a particular situation or project.

4.6 Tools

The tools aspect provides two guiding vision models: an agile tools model and an agile workspace choreography model. Responsibilities and accountabilities (people or teams) are associated with these models at the time of model-enactment and instantiation for a specific situation or project.

4.6.1 Tools Model

The tools model characterizes agile software development tools for the purpose of guiding self-organizing agile teams to use or abandon any specific agile development tool for a particular situation or project. The tools model presents a mental-model to represent various types of tools such as: development tools, automated testing tools, deployment tools, automatic documentation generator tools, communication tools etc. (Table 4).

4.6.2 Workspace Choreography Model

The workspace choreography model is a mental-model or meta-model to specify an agile software development environment or agile workspace for a particular situation or project. A Grid technology-enabled advanced visualization, interaction and collaborative environment (Childers, Disz, Hereld, Hudson, Judson, Olson, Papka, Paris and Stevens 1999) can be used for the construction of an economical agile workspace to support agile software development. We intend to further evaluate this model in our future research.

Guiding Vision: Tools Model
1. ID & Name: The unique ID and name of the tool.
2. Tool Usability: The related details and purpose of the tool
3. Abstraction: The particular supported abstraction mechanism (object, agent, service etc.).
4. People: The requirement of the skilful people to successfully use the tool.
5. Development Style: The supported development style (iterative, rapid etc.).
6. Physical Environment: The supported development environment (co-located or distributed).
7. Constraints and Risks: The possible associated constraints and risks.
8. Business Value: The business value that is added by the tool.
9. Alerts: Any special situations when the tool should not be used.

Table 4. Tools Model

5 Conclusions and Future Work

In this paper we have presented a set of six aspects of an agile software development methodology as a guiding vision model. The purpose of the guiding vision model is to guide the behaviours of self-organizing and empowered agile teams with a cohesive set

of shared information. Agile teams work in an adaptive environment and make decisions to handle day-to-day problems. Therefore, it is necessary to equip agile teams with an appropriate level of information to help them to make informed and rational decisions for delivering optimum business value. The guiding vision model(s) should be updated and refined, as required.

There are a number of mental-models for various abstraction mechanisms such as for object, agent and service-oriented but a standard mental-model for m-abstraction (multi-abstraction) has not yet been developed and reported. In future, we intend to pursue the development of a standard mental-model for agent service-oriented abstraction mechanism (m-abstraction: hybrid abstraction).

Another future line of research is to investigate to what extent this new mental-model of agile development is supported by existing methodology metamodel standards. Rigid metamodels focussed primarily on object-oriented process, such as SPEM (OMG 2002) and OPF (Firesmith and Henderson-Sellers 2002), are not readily extensible to the realm of agile methods. The newer metamodels, however, such as the ISO/IEC (2007) 24744 standard, provide an easily used, inbuilt extension mechanism (Gonzalez-Perez and Henderson-Sellers, 2006) that should be useful in ensuring adequate support should such extensions be found necessary.

References

- AgileManifesto. 2006. Manifesto for Agile Software Development.
- Arsanjani, A. 2004. Service-oriented modeling and architecture. <http://www-128.ibm.com/>
- Augustine, S. 2005. Managing Agile Projects. Pearson Education Inc. USA
- Augustine, S. and Woodcock, S. 2003. Agile Project Management. CC Pace Systems, www.ccpace.com
- Australian Standards.2004. Standard metamodel for software development methodologies. www.standards.com.au/, AS 4651
- Beck K. 2000. Extreme Programming Explained, Addison-Wesley Pearson Education, Boston.
- Cao, L., Zhang, C., and Ni, J. 2005. 'Agent Service-Oriented Architectural Design of Open Complex Agent Systems'. Proceedings of the 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IEEE
- Childers, L., Disz, T., Hereld, M., Hudson, R., Judson I., Olson R., Papka M.E., Paris J. and Stevens R. 1999. 'ActiveSpaces on the Grid: The Construction of Advanced Visualization and Interaction Environments'. ParallelDataCenter Kungl Tekniska Högskolan Seventh Annual Conference (Simulation and Visualization on the Grid), Springer-Verlag, Stockholm, Sweden
- CMMI. 2006. Carnegie Mellon, Software Engineering Institute. www.sei.cmu.edu/cmmi/
- Cossentino, M.2005. From Requirements to Code with the PASSI Methodology (ed. Henderson-Sellers, B. and Giorgini, P.). Hershey, PA, USA, Idea Group Inc.
- DSDM. 2003. DSDM Consortium, Dynamic Systems Development Method Ltd.
- Firesmith, D. G. and Henderson-Sellers, B. 2002. The OPEN Process Framework. Pearson Education, London
- Giunchiglia, F., Mylopoulos J., and Perini, A. 2002. 'The Tropos Software Development Methodology: Processes, Models and Diagrams'. Proceedings of the First International Joint Conference on Autonomous Agents and Multi Agent Systems, Bologna, Italy, ACM Press, New York
- Glaser, B. G. and Strauss, A. L. 1967. The Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine Publishing Company, New York
- Gonzalez-Perez, C. and Henderson-Sellers, B., 2006, On the ease of extending a powertype-based methodology metamodel, keynote paper in *Meta-Modelling and Ontologies. Proceedings of the 2nd Workshop on Meta-Modelling, WoMM 2006*, LNI Volume P-96, 11-25
- Highsmith J.A.I. 2000. Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. Dorset House Publishing, New York.

- ISO/IEC. 2007. Software Engineering - Metamodel for Development Methodologies. ISO/IEC Standard 24744
- Juan, T., Pearce A., and Sterling, L. 2002. 'ROADMAP: extending the Gaia methodology for complex open systems'. Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, Bologna, Italy, ACM Press, New York, NY, USA
- Kumar, K. and Welke, R. J. 1992. Method Engineering: A Proposal for Situation-specific Methodology Construction. Systems Analysis and Design: A Research Agenda, John Wiley and Sons
- OMG, 2002, *Software Process Engineering Metamodel Specification*, OMG document formal/02-11-14
- Padgham, L. and Winikoff, M. 2004. Developing Intelligent Agent Systems. John Wiley & Sons
- Palmer S.R and Felsing J.M. 2002. A Practical Guide to Feature-Driven Development. Prentice-Hall Inc, Upper Saddle River.
- Pandit, N.R. 1996. The Creation of Theory: A Recent Application of the Grounded Theory Method. <http://www.nova.edu/ssss/QR/QR2-4/pandit.html>,
- Qumer A and Henderson-Sellers B. 2006a. 'Measuring agility and adoptability of agile methods: A 4-Dimensional Analytical Tool'. Procs. IADIS International Conference Applied Computing 2006, IADIS Press, 503-507.
- Qumer, A. and Henderson-Sellers, B. 2006b. 'A Framework to Support Non-Fragile Agile Agent-Oriented Software Development'. Proceedings of the International Conference on new Software Methodologies, Tools and Techniques (SoMeT2006), Quebec, Canada (eds. H. Fujita and M. Mejri), IOS Press, 84-100
- XXXX, A. and XXXXX, B., 2007a, An agile software solution framework: support for multi-abstraction mechanisms, submitted to PROFES 2007, Riga
- XXXXX, A. and XXXXX, B., 2007b. An agile toolkit to support agent-oriented and service-oriented computing mechanisms, submitted to PROFES 2007, Riga
- Schwaber K and Beedle M. 2002. Agile Software Development with SCRUM. Prentice Hall.
- SPICE.2003. Software Process Improvement and capability dEtermination for Object/Component Based Software Development. Information Society Technologies (IST) Programme
- Strauss, A. and Corbin, J.1990. Basics of Qualitative Research: Grounded Theory, Procedures and Techniques. Sage Publications, Newbury Park, CA.
- Wood, M. F. and DeLoach, S. A. 2000. 'An overview of the multiagent systems engineering methodology'. First international workshop, AOSE 2000 on Agent-oriented software engineering, Limerick, Ireland, Springer-Verlag, New York
- Wooldridge, M., Jennings, N.R. and Kinny, D. 2000. 'The Gaia Methodology for Agent-Oriented Analysis and Design'. Autonomous Agents and Multi-Agent Systems 285-312