Open access • Book Chapter • DOI:10.1007/978-3-319-19578-0_48

An Approach to Integrating Aspects in Agile Development — Source link

Tadjer Houda, Meslati Djamel

Institutions: University of Annaba

Published on: 20 May 2015 - Computer Science and its Applications

Topics: Agile usability engineering, Agile Unified Process, Lean software development, Empirical process (process control model) and Agile software development

An Approach to Integrating Aspects in Agile Development

Tadjer Houda, Meslati Djamel

An Approach to integrating Aspects in Agile Development

Tadjer Houda [1], Meslati Djamel [2]

[1] LabSTIC Laboratory,
University of 08 Mai 45, Guelma, Algeria
tadjerh@yahoo.fr
[2] Computer Science Department, LISCO Laboratory,
Badji Mokhtar-Annaba University, Annaba, Algeria
meslati_djamel@yahoo.com

Abstract. Separation of concerns is an important principle that helps to improve reusability and simplify evolution. The crosscutting concerns like security, and many others, often exist before implementation, in both the analysis and design phases, it is therefore worthwhile to develop aspects oriented software development approaches to handle properly the concerns and ensure their separation.
Moreover agile methods attempt to reduce risk and maximize productivity by carrying out software development with short iterations while limiting the importance of secondary or temporary artifacts, however these approaches have problems dealing with the crosscutting nature of some stakeholders' requirements. The work presented in this paper aims at enriching the agile development using aspect oriented approaches. By taking into account the crosscutting nature of some stakeholders' requirements, the combination of the two approaches improves the software changeability during the repeated agile iterations.

Keywords: aspect oriented, Constraints, Extreme programming, Separation of concerns, User stories.

1 Introduction

Taking into account the concerns in the analysis phase is currently regarded as an important step that could have a positive impact on subsequent development phases and, consequently, there are several Aspect Oriented Requirement Engineering models (AORE models) such as MC AORE model [13], Quality AORE model [10], Vgraph model [16] and Theme/doc model [2].

Moreover agile methods have become, due to their pragmatism, favoured approaches for complex systems development. The use of the early separation of concerns in agile approaches is an important issue which can cause considerable fallout in terms of software development management and clear architectural structuring. We found in literature several agile approaches: Extreme Programming [8], Scrum [14], Feature-Driven Development (FDD) [11], and Dynamic Systems Development Method (DSDM) [15], Crystal Methodologies [4], Adaptive Software Development (ASD) [5]. Combining aspects oriented approaches with these agile approaches, eliminates the tangling and the scattering of the code they produce and consequently reduces the effort of understanding and changing this code during the repeated agile iterations.

In this paper, we present a work which aims at combining the separation of concerns, requirements' engineering and the well known Extreme Programming

approach (Xp), in order to achieve a synergy that enhances the Xp approach of development by a convenient handling of concerns. This work focuses on how aspect concepts can be integrated within the requirement level of the agile development context and particularly in the Xp approach.
In the rest of this paper, we present, briefly, the Xp approach, the aspect oriented requirements engineering. Thereafter, we explain the proposed combination. Then we describe some related work and, finally, we give a conclusion.

2 Agile approaches

The agile approaches are a family of pragmatic development approaches built to deliver products on time, budget and with high quality. These approaches focus on strong customers' involvement and guarantee that they will be satisfied by their project. The Xp approach is one of the most commonly used among agile approaches [7]. The Xp approach provides a life cycle model of software which is used as a guide for organizing the development team. This model is presented in Figure1. User stories are an important concept in the Xp development and are usually the starting point of all the Xp processes. By choosing them, the customer kick starts the iteration process. They represent what the customer wants the system to do [8]. The system development is a succession of such iterations where the requirements are continuously being defined by means of user stories. These user stories should feed into the release-planning meeting and to the creation of the user acceptance tests.

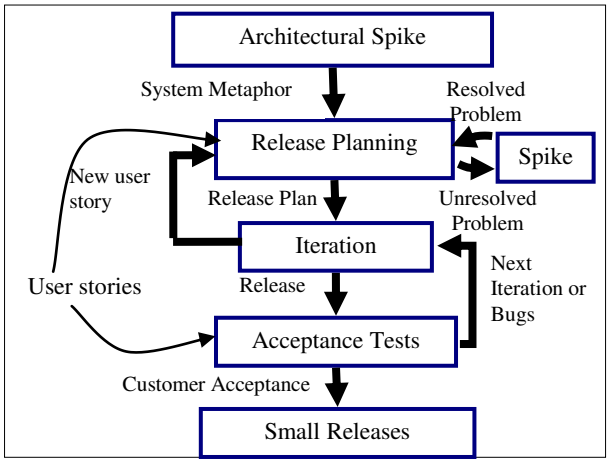


Fig. 1. Lifecycle of project Xp [7]

A release-planning meeting is used to create the release plan, which lays out the overall project. That is, the release plan indicates which user stories will be implemented and in which release this will happen. It also indicates how many iterations are planned and when each iteration will be delivered.

At the beginning of each iteration, an Iteration Planning meeting is held to determine exactly what will happen within that iteration. Such just-in-time planning is considered an easy way to stay on top of changing user requirements.
The acceptance tests are created from the user stories, written at the start of the project. Each iteration implements one or more user stories; these stories will be translated into a series of acceptance tests during the iteration.

3 Aspect Oriented software development (AOSD)

In object orientation, applications are modeled and implemented by decomposition of both the problem and solution space into objects, where each object embodies a single concern. However, some concerns still remain scattered throughout many different objects because they don't naturally fit within object boundaries. Such concerns (such as security, mobility, distribution, and logging) crosscut the other concerns. Aspect-oriented programming provides an elegant solution to this problem. Initially, the concept of aspect has been introduced in the context of programming. However, its use has become widespread to cover, among other things, analysis, design and evolution of applications.
To improve the software development, we need fully aspect oriented approaches that support aspects ranging from the analysis phase till implementation and testing. These approaches are often described as Aspect Oriented Software Development (AOSD) and encompass a range of techniques to achieve a better modularity.

Aspect-Oriented Software Development (AOSD) is regarded as a promising method that allows systematic identification, modularization, representation, and composition of such crosscutting concerns. Currently, it is commonly accepted that a good separation of concerns improves system modularity, reduces the complexity of software systems and the tangle of their code, facilitates reuse, improves comprehension, simplifies the integration of components and decreases the change, reducing the cost of adaptation, evolution and maintenance.

The requirement engineering is of vital importance because of its influence on the rest of the development. This is a starting point for many researches that aim at improving the separation of concerns at the requirements' level.
The motivation of these researches lies in reducing the cost of adaptation, evolution and maintenance. According to [3] the main activities in the requirements are: identification, capture, composition and analysis. Currently, several AORE models have been proposed and used as mentioned previously.

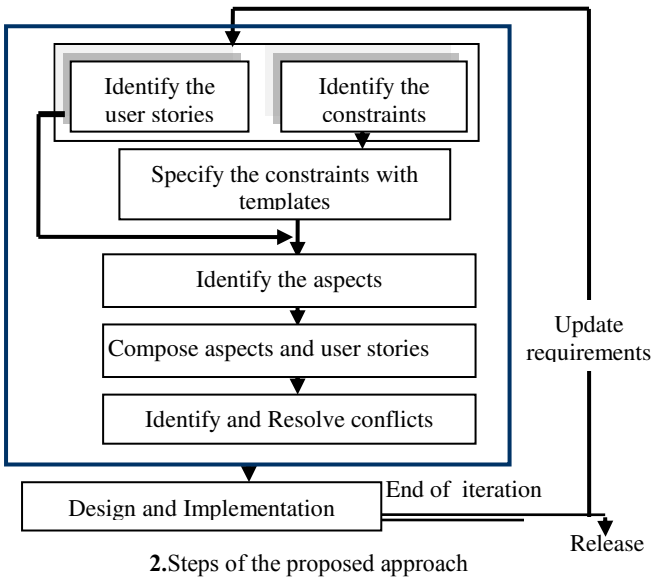
4 The proposed approach

Our work consists of integrating the aspect oriented approach in the Extreme programming approach and focuses particularly on the analysis phase. Our goal is

to incorporate the concepts of AORE in the Xp process to make it more efficient and improve the productivity of the development team.
AORE's goal is the separation of concerns at the level of requirements, which may influence the way of using Xp in a project development. The agile development in general and Xp in particular can benefit from the AORE. In the Xp approach, the user stories represent the requirements of a system in the sense that the requirements are informally described by these stories from the customers. That is why our approach is mainly based on the user stories.

4.1 Steps of the proposed approach

Figure 2 summarizes the main steps we propose in our approach. We describe them shortly in the following.
Step 1: Identify user stories and constraints. In this stage customers and the development team meet to discuss the main functionalities to be achieved by the system. These functionalities are written by customers in the form of user stories on indexed cards. Each story has a short name that describes the functionality. Other details are added such as the risk to improve planning and performance of iteration. At the end of this stage, the customer must choose the stories to implement in the current iteration. Non functional requirements can address a variety of system needs and they can be considered as constraints on the system's behavior. Thus the customer must identify these constraints.



2.Steps of the proposed approach

Step 2: Specify the constraints. Based on the approach described in [10] the constraints are specified using templates as shown in table1.
Each constraint is defined as follows:

Table 1. Template for constraint

Constraint	Description
Name	Name of constraint.
Description	Description of constraint.
Influence	Lists of user stories affected by this constraint.
Priority	Priority assigned for this constraint.
Contribution	Represents how a constraint can be affected by other constraint. This contribution can be negative (-) or positive (+).

Step 3: Identify aspects. If a constraint affects several user stories then this constraint is an aspect (taking in to account the information in row influence). In other words, if a constraint is triggered from several other user stories so it is considered as an aspect.

Step 4: Compose aspects and user stories. In this stage we try to compose crosscutting constraints and user stories in order to find the impact of an aspect on the requirements. We must define composition rules showing how an aspect influences behavior of a set of user stories.

Step 5: Identify and Resolve conflicts. Identification of conflicts is based on the MCAORE technique [13] which uses a contribution matrix where each aspect may contribute negatively (-) or positively (+) to the others. If aspects have the same priority and contribute negatively then these aspects are in conflicts.

The conflicts in our approach are resolved through effective negotiation with customer who is part of the development team (on site customer).

Step 6: Design and Implementation. Xp like the other agile processes prioritizes pragmatic design for long-team change. The final set of user stories and aspects plus composition rules are used in the implementation, so an aspect oriented programming language could be used.

At the end of an iteration, the customer can check the product by the acceptance tests to detect errors or add other features. Following the change requirements, it is necessary to repeat the separation of concerns again.

4.2 Example

In this section we apply the previous steps to the creation of a website for the company SOUTH COAST NAUTICAL SUPPLIES to augment their print catalog. This example is taken from [5].

Step 1. Identify user stories and constraints. These are the user stories and constraints which are written by the stakeholders:

- User story 1: A user can do a basic simple search that searches for a word or phrase in both the author and title fields.

• User story 2: A user can put books into a "shopping cart" and buy them when he is done shopping.
• User story 3: To buy a book the user enters her billing address, the shipping address and credit card information.
• User story 4: A user can establish an account that remembers shipping and billing information.
• User story 5: A user must be properly authenticated before viewing reports.
• User story 6: An administrator can add new books to the site.
• User story 7: An administrator can delete a book.
• User story 8: An administrator can edit the information about an existing book.

Constraint C1: The system must support peak usage of up to 50 concurrent users.
Constraint C2: For audit purposes, all transactions in the system have to be kept.

In this iteration the first constraint shows that the system must support concurrent manipulation by at least 50 users which implies that there is a *multiple access* system. The second constraint is *audit* means that the action at each step is recorded.

By analysis of the stories identified for this iteration we can extract other constraints which are not written by the stakeholders. For example: User story 5, in this case the security must be guaranteed as the information provided by the user is personal data. The constraint identified here is *security*.
In the same way developers can also identify another constraint is the *login*.

Step 2. Specify the constraints.

Specification is as follows:

Table 2. Template for C1

Name	Multiple access
Description	Multiple users can use the system simultaneously.
Influence	multiple user stories
Priority	must have
Contribution	(+) Audit, (+) login

Table 3. Template for C2

Name	Audit
Description	The action at each step is captured and kept
Influence	multiple user stories
Priority	must have
Contribution	(+) Security, (+) multiple Access, (+) login

Table 4. Template for C3

Name	Security
Description	Only authorized users can access information.
Influence	multiple user stories
Priority	must have
Contribution	(+) Audit, (+) login

Table 5. Template for C4

Name	Login
Description	Provides the ability to connect and disconnect.
Influence	multiple user stories
Priority	must have
Contribution	(+) Audit, (+) Security, (+) multiple Access

Step 3. Identify aspects. From the table 2 to table 5, we deduce the following constraints:
Multiple access, A*udit, security and login* affect multiple user stories. So, these constraints are crosscutting and therefore represent aspects.

- Aspect1: *Multiple access*
- Aspect2: A*udit*
- Aspect3: *Security*
- Aspect4: *Login*

Step 4. Compose aspects and user stories. To combine aspects with the basic user stories we will first define composition rules indicating how these aspects influence the behavior of a set of basic user stories.
An example of composition rules in the case presented above is:

For safety, there is a recovery situation "Overlap", represented by the qualifiers before or after. Security for User story 5 is applied before viewing the report because we are in front of a protection in this case the composition rule will as follows:

- Security. Overlap. Before user story 5.

For the audit, there is also a situation of "Overlap", represented by before or after. The audit here is applied after each story affected recorded for each action, in this case the composition rule is as follows:
- Audit. Overlap. After all history

Step 5. Identify and Resolve conflicts. The contribution matrix which is symmetrical indicates whether aspects contribute positively or negatively. In our example aspects contribute positively and in this case no conflict appears in this iteration.

Table 6. The contribution matrix

Aspects	Audit	Security	Multiple Access	Login
Aspects				
Audit		+	+	+
Security				+
Multiple Access				+
Login				

Step 6. Design and Implementation.

At the end of this iteration, suppose that the customer wants to add other constraints.

A second iteration is then necessary. We describe it in what follows.

Step 1. Identify new user stories and constraints. These are user stories and constraints which are written by the customers:

- User story 9: A user can search for books by entering values in any combination of author, title and ISBN.
- User story 10: A user can view detailed information on a book. For example, number of pages, publication date and a brief description.
- User story 11: A user can put books into a "wish list" that is visible to other site visitors.
- User story 12: A user, especially a Non-Sailing Gift Buyer, can search for a wish list based on its owner's name and state.
- User story 13: A user can check the status of her recent orders.
- User story 14: If an order hasn't shipped, a user can add or remove books, change the shipping method, the delivery address and the credit card.
- User story 15: A user can view a history of all of his past orders.
- User story 16: A user can easily re-purchase items when viewing past orders.
- User story 17: A user can see what books we recommend on a variety of topics.
- User story 18: A user can remove books from her cart before completing an order.

Constraint C5: A customer must be able to find one book and complete an order in less than 90 seconds.

Step 2. Specify constraints.
For the next iteration, and due to changes in the requirements, a new specification is necessary (Table 7 to 11).

Table 7. Template for C1

Name	Multiple access
Description	Multiple users can use the system simultaneously.
Influence	multiple user stories
Priority	must have
Contribution	(+) Audit, (+) login, (-)Response time

Specification is as follows:

Table 8. Template for C2

Name	Audit
Description	The action at each step is captured and kept
Influence	multiple user stories
Priority	must have
Contribution	(+) Security, (+) multiple Access, (+) login, (-)Response time

Table 9. Template for C3

Name	Security
Description	Only authorized users can access information.
Influence	multiple user stories
Priority	must have
Contribution	(+) Audit, (+) login, (-)Response time

Table 10. Template for C4

Name	Login
Description	Provides the ability to connect and disconnect.
Influence	multiple user stories
Priority	must have
Contribution	(+)Audit, (+)Security, (+)multiple Access, (-)Response time

Table 11. Template for C5

Name	Response time
Description	Period of time in which the system responds to a service
Influence	multiple user stories
Priority	must have
Contribution	(-) Security, (-)multiple Access, (-) Audit, (-) login

Step 3. Identify aspects. From constraints which affect more than one user story, we deduce the following:
The first four (*Multiple access,* A*udit, security and login)* are already identified as aspects in the first iteration, the second constraint (response time) is crosscutting as they affect multiple user stories (taking into account the information in row influence) and therefore represent aspect. So for this iteration we add a new aspect:

- Aspect5: Response time

Step 4. Composed aspects and user stories. To combine the new set of aspects with the basic user stories we will define new composition rules indicating how these aspects influence the behavior of a set of user stories.

In this step we try to dial the new aspect 'response time' identified in the previous step with the stories it affects. As this aspect is required in parallel with the stories he forced this leads to use the relation "wrap".

For other aspects just add composition links with the stories of this iteration.

Step 5. Identify and Resolve conflicts. In our example the contributions between aspects are presented in the following table:

Table 12. The contribution matrix

Aspects Aspects	Audit	Security	Multiple Access	Login	Response time
Audit		+	+	+	-
Security				+	-
Multiple Access				+	-
Login					-
Response time					

This table indicates the presence of conflicts between some aspects, if these aspects apply to the same user stories with the same priority. For example security and response time contribute negatively to each other. They constrain each other's behavior and have the same priority and apply to the the same user story, thus a conflict is arise.

As in this iteration we are faced with a conflict, it is resolved by negotiations between the customer and the development team.

Step 6. Design and Implementation.

The iterations are repeated until there is no need to add or update requirements.

5 Related wok

Several approaches are intended to identify crosscutting concerns during the early stages of development [9]. Aspect Oriented Requirement Engineering (AORE), described in [13], proposes a model for Aspect Oriented Requirement Engineering that supports the separation of crosscutting properties at the requirements level. Concerns and associated requirements are identified from different viewpoints. The rules of composition are defined using XML. In [10], functional requirements are specified using use cases based approach. The quality attributes are detailed extensively in a template, which among other details also lists down the decomposition of the quality attribute, priorities (max, high, low, and min), and influence of the quality attribute. By observing the influence of a quality attribute and associated requirements, crosscutting concerns (quality attributes) are identified. Here, a set of UML models are integrated to the crosscutting quality

attributes. Baniassad and Clarke [2] propose the Theme approach that does not identify the crosscutting concerns from traditional requirements engineering approaches. They introduced the concept of action view, clipped action view, base themes, and crosscutting themes to provide support for the aspect orientation in the analysis and design. Theme supports activities of requirements analyses. The results of analyses are mapped to the UML models. Despite the diversity of these approaches, no one among them takes into account the agile development. the FDD approach takes into consideration the integration of aspects in a contextual agile development [12]. In [1], a method is proposed for unifying agile and AO requirements analysis approaches.

6 Conclusion

The Xp approach is a development approach that can produce quickly software of high quality. This development approach may benefit from aspect-oriented requirements engineering approaches in a variety of ways.
The work presented in this article is a proposition of integration between separation of concerns and requirements engineering in an agile development context and particularly in the extreme programming approach. The main contribution of our work is its focus on the user stories and constraints as the starting point of the integration. Our approach is still at its beginning and we are now using it for more complex systems.

7 References

1 Araujo J and J.C. Ribeiro, "A scenario and aspect-oriented requirements agile approach," International Journal of Computing Science and Applications, Vol. 5, No. 3b, pp. 69-92 (2008)
2 Baniassad. E and Clarke. S. Aspect-oriented analysis and design: Theme Approach, Addison Wesley professional (2005)
3 Baniassad. E, P.c.Clements, J.Araujo and. Moreira. A, Rashid. A, B.Tekierdogan, "discovering Early aspects", IEEE Software January/February (2006)
4 Cockburn .A, Crystal methodologies: The Cooperative Game, Addison-Wesley (2006)
5 Cohn.M, User Stories Applied: For Agile Software Development, Addison Wesley, March (2004)
6 Highsmith.J, Adaptive Software Development, Dorset House, New York, NY (2000)
7 Hunt.J, Agile software construction (2006)

8 Kent.B, Extreme Programming Explained: Embrace Change, Addison Wesley (2000)
9 Moreira, A. Araújo, J., The Need for Early Aspects, LNCS 6491, Springer-Verlag Berlin Heidelberg , pp. 386-407 (2011)
10 Moreria.A, Araújo .J, and Brito.I. "Crosscutting quality attributes for requirements engineering". SEKE2002: Fourteenth International Conference on Software Engineering and Knowledge Engineering, Ischia, Italy, 15-19 July (2002)
11 Palmer.S. R and Felsing J. M, A Practical Guide to Feature-Driven Development, Addison-Wesley (2002)
12 Pang.J and Blair.L, "Refining feature driven development - a methodology for early aspects", Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design (2004)
13 Rashid.A, Moreira A., and Araújo.J, "Modularization and composition of Aspectual Requirements", 2nd International Conference on Aspect-Oriented Software Development. Pages 11-20 Boston (2003)
14 Schwaber.K et Beedle.M, Scrum: Agile Software Development, Prentice-Hall (2002)
15 Stapleton.J, Dynamic Systems Development Method:The method in practice, Addison-Wesley (1997)
16 Yu. Y, Leite. J. C. S, and Mylopoulos.J, "From goals to aspects: discovering Aspects from requirements goal models", The 12th IEEE International Requirements Engineering Conference, Kyoto, Japan (2004)