A Comparative Study of Aspect-Oriented Requirements Engineering Approaches

Américo Sampaio, Phil Greenwood, Alessandro F. Garcia and Awais Rashid
Infolab21, Computing Department, Lancaster University, Lancaster, UK
{a.sampaio, greenwop, garciaa, marash}@comp.lancs.ac.uk

Abstract

Aspect-Oriented Requirements Engineering (AORE) aims at improving separation of concerns in the problem space by offering new ways of modularising requirements. Over recent years several AORE approaches have emerged by evolving contemporary requirements approaches such as viewpoints-, scenarios- and goal-based models. Due to the novelty of these techniques, there is a lack of systematic comparative studies analyzing the benefits and drawbacks they can offer to the requirements engineering practice. This paper presents a case study contrasting four eminent AORE approaches in terms of time effectiveness and accuracy of their produced outcome. We address challenges related to the heterogeneous definitions for AORE model concepts as well as the fact that they perform similar general requirements process activities in different ways. In order to address these challenges, we provide a mapping of the AORE approaches onto general RE activities and provide a common naming scheme. The case study results show that specification of aspect compositions in AORE presents an effort bottleneck that has to be carefully weighed against the added benefits of modularity and analysis of systemic properties offered by AORE. Consequently, our study provides an initial yet significant stepping stone towards improving the evaluation of AORE approaches and understanding their contribution to requirements engineering.

1. Introduction

Requirements Engineering (RE) encompasses several important activities [1, 2] of the software engineering lifecycle such as requirements elicitation, analysis, specification, conflict resolution and validation. The main goal of RE is to clearly specify stakeholders' requirements enabling the software engineers to gain a deeper understanding about the functionalities, restrictions and properties of the system to be developed as well as the environment in which the system will operate.

Over recent years several researchers [3-7] developed Aspect-Oriented Requirements Engineering (AORE) approaches. The aim is to improve separation of concerns at the requirements level by offering new ways of modularising systemic requirements in units called *early aspects*. Such requirements are otherwise scattered over and tangled with various requirements units (e.g., viewpoints, use-cases, etc.) similar to the problems tackled by the aspect-oriented programming community [8]. AORE approaches have been defined either by evolving contemporary requirements approaches such as viewpoints- [4, 5], scenarios- [6] and goal-based [7] models or by developing new approaches [3, 9].

The proposed solution to this requirements modularity problem is to: (i) separate concerns that impact several other modules, including broadly-scoped non-functional requirements such as security, safety and performance, into a single module, and (ii) specify how this module (i.e., the early aspect) constrains and affects the others (see Figure 1). This improved modularity is expected to bring benefits [3-6, 10] such as improving management of change impact. Hence it is expected that when a crosscutting property needs to be modified the respective changes will be contained in one single place and the potential influence of these changes can also be propagated in a modular fashion through the aspect composition specifications.

Even though some of these claims have been partially investigated in [3-6, 10, 11], they were based on relatively simple examples, and carried out in isolation. In fact, we are not aware of any empirical study conducted in the field that investigates a systematic way for assessing and comparing the effectiveness of these approaches. In addition, there is no empirical evidence of the benefits and drawbacks of AORE to the current requirements engineering practice. There is no evaluation framework for the field which impedes the development and execution of such empirical case studies.

This paper presents a case study that compares four different AORE approaches in terms of time-effectiveness of their various activities measured by effort data in person-minutes. Also, the quality of their produced outcomes is measured by precision

and recall data of the concepts present in each requirements specification produced using the different AORE approaches. The goals of our case study were fourfold (Section 4):

Q1. Which activities are the main bottlenecks in terms of effort for each AORE approach?
Q2. Is there any specific approach that significantly differs from the others in terms of effort?
Q3. What factors mostly contribute to differences in time-effectiveness?
Q4. Do the AORE specifications produced by each approach have comparable quality?

A major challenge in the case study was dealing with the heterogeneous definitions of AORE model concepts as well as the fact that they perform similar general requirements process activities in different ways. In order to standardize data collection and also normalize the gathered data we provide a mapping of the AORE approaches onto general RE activities and provide a common naming scheme.

The effort and quality data gathered by our case study presents interesting results. For example the effort data shows that the specific AORE activity of composition specification (not present in traditional RE approaches) is one of the most time consuming. Also interesting to note is that the different AORE models produced were of comparable quality even though the effort was varied (Section 4).

The remainder of this paper is structured as follows. Section 2 presents the necessary background on AORE approaches and their comparison challenges. Section 3 describes the common naming scheme and common process scheme. Section 4 describes the case study comparing four different AORE approaches in terms of effort (measured in person-minutes) and quality of the outcome (measured in terms of accuracy metrics of precision and recall of identified concepts). Section 5 discusses some limitations of the conducted case study and some relevant findings. Section 6 describes some related works and section 7 concludes the paper.

2. Comparing AORE Approaches

Evaluating and comparing AORE approaches are daunting tasks due to variations among them. Examples of these variations are terminology used, concepts used in the different AORE models, the way the modularization units are structured and so on. In order to understand the differences we will first present an overview of some contemporary AORE approaches. Our goal is to stress their commonalities and differences that motivated the need for creating the common naming and common process scheme for our case study. For a more detailed description of AORE approaches see [12].

One fundamental difference among AORE approaches is centred on the way the modularization units are structured. All approaches recognise the concept of an early aspect even though this concept can be named differently depending on the approach. An early aspect as shown in Figure 1 can be understood as an abstraction that modularizes the requirements of the same crosscutting concern (e.g., Security, Performance, Logging, Add Item to Shopping Cart, Sign-in User, etc.) that influence or constrain other modules of the requirements model.

When the requirements engineer is building the requirements specification, the model is then structured differently depending on the AORE approach in use. Some approaches [3-7], called asymmetric, provide a clear separation of what are the base and crosscutting abstractions. For example in [4, 5] viewpoints are base abstractions while aspects are broadly scoped non-functional properties that crosscut several viewpoints. In a similar fashion, scenario-based AORE [6] makes a distinction between base scenarios and crosscutting ones (called aspectual scenarios). Goal-based AORE also makes a distinction between base abstractions (NFR goals and decomposed softgoals and tasks) and aspectual requirements. Another approach [3] decomposes the requirements into units called themes and makes a distinction between crosscutting and base themes.

On the other hand symmetric approaches [10, 11] treat the decomposition units uniformly and consider everything to be a *concern*. These concerns can have a functional or non-functional nature and their crosscutting behaviour, if present, is represented in the compositions. In addition to the above base-aspect and uniform concern treatment dichotomy, the concept of what an early aspect is varies from an approach to another. For example, some approaches only consider early aspects to be non-functional requirements while others consider functional early aspects as well. Therefore, it is vital to have a *common naming scheme* to address this.

Another dimension in which the approaches are similar but also contain slight variations is the set of process activities that comprise each approach. Some common activities present in all approaches are:

- *Identification:* for discovering the decomposition units (e.g., base and crosscutting concerns);
- *Composition:* for specifying how the concerns are composed (e.g., how the early aspects affect the base concerns and in which requirements);
- *Conflict Resolution*: for investigating the mutual influences of different concerns. For example, the encryption needs imposed by a security aspect can negatively contribute to the real-time performance of the system.

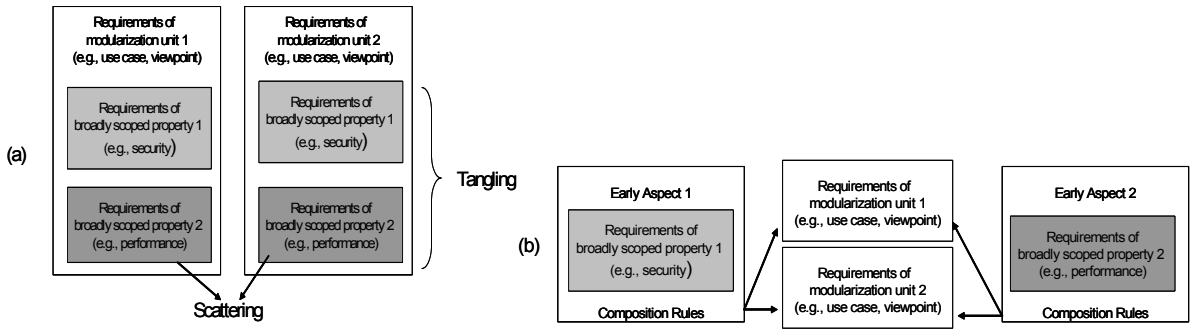

Figure 1 (a) non-AORE approach containing scattering and tangling. (b) AORE approach with early aspects for better modularisation

These activities are conducted in different ways by different approaches as they deal with different models and sometimes require different sub-steps to realise the same general activities. As an example in the *Conflict Resolution* activity for viewpoint-based AORE [4, 5] a contribution matrix is built to analyse the influences of different concerns on the viewpoints. On the other hand, for NFR-based AORE, the NFR catalogues and goal decompositions are analysed to check for intra-goal trade-offs.

If one considers evaluating different AORE approaches for the sake of comparing effort expended in the various activities as well as effectiveness of such activities, it is necessary to have a *common process scheme* so that one can collect measures (e.g., effort in person-minutes) for comparative analysis.

3. Common Schemes

This section discusses in more details the elements of common naming scheme and common process scheme. The goal of the *common naming scheme* is to have a uniform terminology scheme for the different types of decomposition units in the various AORE approaches. This scheme subsumes the different types of concepts present in the various AORE approaches, which are:

- ***Functional Concern***: modularization unit that groups functional requirements that do not represent crosscutting behaviour. These are also called base abstractions in some asymmetric AORE approaches such as viewpoints- and scenario-based AORE. In symmetric approaches these can be considered concerns of a functional nature that do not constrain multiple modules, i.e., functional concerns of a non-systemic nature.
- ***Functional Early Aspect***: modularization unit that groups functional requirements that constrain multiple other modularization units (crosscutting behaviour). The way that the functional early aspect constrains each module can vary and be specified in compositions. These types represent functional early aspects in asymmetric approaches and functional concerns in symmetric approaches that constrain multiple other modules.
- ***Non-Functional Early Aspect:*** By default non-functional requirements are imposed over several requirements of the system so they naturally have crosscutting behaviour and can be grouped in a modularization unit as well. Also, the way the constraints are imposed over other modules can vary and be specified using composition rules. They represent non-functional early aspects in asymmetric approaches and non-functional concerns in symmetric approaches.

These concepts represent our common naming scheme for the various types of concerns in AORE approaches. This will be important for our accuracy evaluation detailed in Section 4.

The *common process scheme* was first idealized before we conducted the case study presented in Section 4. We felt the necessity of having a common process so that we could investigate the effort spent in each activity when using different approaches. The common process scheme denotes a set of activities that are common across several AORE approaches. The process scheme does not address all requirements engineering activities but focuses mainly on *identification of model concepts*; *structuring the requirements specification*; and *conflict resolution* as shown in Figure 2. These activities are general to all requirements engineering approaches, even non-AORE ones, and represent the common set of activities pertaining to the RE lifecycle in which the AORE approaches vary.

Table 1 shows how these general activities are executed by specific AORE approaches. For example, considering the viewpoints-based AORE approach [4, 5], the *identification* activity is sub-divided into three others pertaining to identification of viewpoints, early aspects and crosscutting relationships. The *structuring* activity is also sub-divided into three others of gathering viewpoints requirements, gathering early aspects requirements and specifying the composition rules. Finally, the *conflict resolution*

activity encompasses sub-activities of building the contribution table for representing the mutual influences of early aspects (e.g., security contributes negatively to performance), assignment of weights to quantitatively assess the degree of the conflict and resolve conflicts to support decision making.

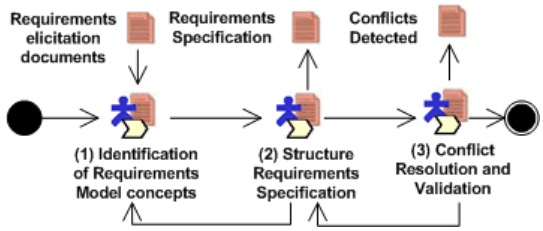

Figure 2 - Common Process Scheme

If one had considered another AORE approach the realization of the *general activities* would be different. For example, the *identification* activity in the scenario-based approach is sub-divided into identification of [6]: non-functional concerns, actors and functional use cases, and candidate aspects in non-functional concerns and use cases.

Table 1 presents a mapping of the activities of the common process scheme in some AORE approaches. We focus on the approaches used in our case study and we also present the effort data collected discussed in Section 4. More details about the approaches can be seen in [4-7, 10, 12].

4. A Real World Case Study

The case study conducted is part of a test-bed project initiative that has the aim of conducting evaluation of aspect-oriented approaches at several lifecycle stages such as requirements, architecture, design, and implementation. The goal of the project is to establish a common ground for researchers in the aspect-oriented field so that they can reuse the results of the project (e.g., evaluation frameworks, metrics, collected data) in other case studies to improve the current state of practice of AOSD and investigate the benefits/drawbacks of such approaches in more detail. Section 4.1 describes our case study whose goal was to compare four different AORE approaches.

4.1 Comparison of AORE approaches

Evaluating software engineering approaches is always a challenging task [13-15] as normally one has to use subjects that participate actively in the study having different backgrounds and experiences that can influence the results obtained.

The goal of our case study was to compare four different AORE approaches with respect to their time-effectiveness (Section 4.1.1) and the quality of their outcome (Section 4.1.2). While time-effectiveness was measured in terms of effort data (person-minutes), the quality of outcome was measured in terms of precision and recall of the produced models.

The four AORE approaches selected for comparison were: Viewpoints-based AORE [4, 5], Multidimensional Separation of Concerns (MDSOC) [10], Aspect-Oriented Requirements Analysis (AORA) [16] and Goal-based AORE [7]. The choice was driven by their maturity and availability of relevant expertise to conduct an AORE analysis.

The case study involved four requirements engineers, each given the responsibility of using one of the approaches to perform the same task. The common task was to restructure an existing document describing a system called Health Watcher [17] which is a real world system used for registering and querying health complaints. The existing available documentation about this system is a 19-page (3900 words) use case description of the system. The Health Watcher system was selected as our case study since the goal of the test-bed project is to conduct evaluation at different lifecycle stages and this system already had a sound object-oriented implementation as well as several interesting crosscutting concerns which made it interesting for investigation of aspect-oriented techniques. Moreover, it also had available requirements documentation, which had been constantly updated and evolved since the system's deployment in 2001.

The task of all the requirements engineers was to restructure the original documentation into a new specification following one of the AORE approaches. Each subject selected for the case study was an expert in one of the AORE approaches (the one s/he used) and all of them had a similar level of expertise as they were all final year PhD students whose PhD topics were related to the approaches they used and whose supervisors were proponents of the approaches. None of the subjects were previously familiar with the Health Watcher system and the only documentation available about the system, the use case specification mentioned above, was used by all subjects. Moreover, during the case study there was no communication among the participants and they were not aware of each other's results.

The common practice in requirements engineering is to first start by eliciting requirements with stakeholders through interviews. In our case study this phase was skipped and none of the subjects had any contact with any other documentation and did not interact with any stakeholders. Even though this was different from common practice, this was a positive point from the perspective of the internal and construct validity [15] as all subjects were restricted by the same boundary conditions (similar expertise, same input, no contact with stakeholders).

	(1) Identification of Requirements model concepts	***(2) Structure requirements specification***	**(3) Conflict Resolution and Validation**
Viewpoint-based AORE **Total: 184 person-minutes***	(1.1) Identification of Viewpoints – **tool support** -*13 minutes* (1.2) Identification of Concerns (early aspects) – **tool support** - *2 minutes* (1.3) Identification of Crosscutting Relationships - **tool support** *6 minutes* **Total: 21 minutes**	*(2.1) Gather Viewpoints requirements – **tool support** - 52 Minutes* *(2.2) Gather Early Aspects requirements – **tool support** -19 minutes* *(2.3) Specify Composition Rules – 38 minutes* **Total: 109 minutes**	(3.1) Build Contribution Table – **partial tool support -** 23 minutes (3.2) Attribute weights to Conflicting Aspects – 13 minutes (3.3) Resolve Conflicts 18 minutes **Total: 54 minutes**
Multidimensional Separation of Concerns (MDSOC) **Total: 312 person-minutes**	(1.1) Identify Concerns minutes - *35 minutes* (1.2) Identify Coarse-grained Concern Relationships - *61 minutes* **Total: 96 Minutes**	*(2.1) Specify concerns – 85 minutes* *(2.2) Specify Concern Projections Using Composition Rules - 68 minutes* **Total: 153 minutes**	(3.1) Build Contribution Table – 19 minutes (3.2) Identify Reflected Projections – 18 minutes (3.3) Attribute weights to conflicting Concerns – 14 minutes (3.4) Resolve Conflicts - 12 minutes **Total: 63 minutes**
Aspect-Oriented Requirements Analysis (AORA) **Total: 521 person-minutes**	(1.1) Identify Concerns – *90 minutes* (1.2) Identify responsibilities - *30 minutes* (1.3) Identify Contributions – *10 minutes* **Total: 130 minutes**	*(2.1) Build concern Models – **tool support** -60 minutes* *(2.2) Compose concerns by specifying the composition rules – **tool support -** 180 minutes* **Total: 240 minutes**	(3.1) Conflict analysis - **tool support** - *150 minutes* **Total: 150 minutes**
Goal-based AORE **Total: 467 person-minutes**	(1.1) Identify goals, softgoals, tasks and relationships – *87 minutes* **Total: 87 minutes**	*(2.1) Build V-graph tree - **tool support** 132 Minutes* *(2.2) specify compositions and crosscutting relationships - 73 minutes* **Total: 205 minutes**	(3.1) analyse trade-offs and early aspects interactions - *175 minutes* **Total: 175 minutes**

Table 1 - Mapping of different AORE approaches to the common process. *All the activities conducted by only one person.

Moreover, this situation is not completely implausible in practice as in some real scenarios the requirements engineers do not have the opportunity to have much contact with the stakeholders, for example, in mass market application development (e.g., web and off-the-shelf software)[18] where the number and diversity of users are usually extremely high. In these cases the requirements engineers have to elicit the requirements based on available documentation such as marketing studies, legacy specifications and user manuals.

The goal of the case study was to gather data (effort, accuracy) to investigate how the different engineers performed on the same task of structuring an AORE specification in their specific techniques. All the subjects collected data while conducting their tasks and this data was later sent to another researcher for analysis purposes. The researcher that received the results was a post-doc in aspect-oriented software development but not linked with any of the analyzed approaches. The main purpose of using an independent researcher to evaluate the data collected was to avoid bias in favour of one approach over the other.

4.1.1 Time-Effectiveness Comparison

As discussed in Section 3 and shown in Table 1 each AORE approach has different activities. This is why the common process scheme of Figure 2(a) is useful for organizing the groups of activities for effort data collection. With this in mind each requirements engineer took notes on the time spent for each activity as shown in Table 2.

date	**Start**	**stop**	**Activity**	**common process activity**	**observ.**
04/Sep /06	18:30	19:15	Identify Concern	Activity 1	Manual

Table 2 – Excerpt from data collection tables

Table 2 shows an excerpt of data collected from the MDSOC approach. It shows part of the time spent for the activity of identifying concerns (45 minutes) that refers to activity 1 of the common framework (Identification of requirements model concepts). The observation column was used by the requirements engineer to describe that the activity was done manually (without tool support). This information is important as some tasks of some approaches were supported by tools. For instance, the identification and structuring tasks of the viewpoints-based AORE

approach were supported by the EA-Miner tool [19, 20].

All subjects were explicitly instructed to collect data while working on the task. They were advised to be strictly honest and cautious to avoid counting times while they were interrupted or doing any other activity which was not related to their case study task.

The goal of this effort comparison was not solely to achieve results such as approach X is more time-effective than approach Y. More importantly, as our research is also related to the improvement of aspect-oriented practices in general, our goal also was to identify possible avenues of improvement of these AORE approaches such as, for example, providing tool support for costly activities.

As we have already developed some tools, such as EA-Miner that provides automated support for identification and structuring activities (activities 1 and 2 of Figure 2(a)) we also wanted to verify if the tool support really helped to reduce the effort spent in those activities. Therefore, the main research questions we aimed to answer with this effort comparison were:

Q1. Which activities are the main bottlenecks in terms of effort for each AORE approach?
Q2: Is there any specific approach that significantly differs from the others in terms of effort?
Q3: What factors mostly contribute to differences in time-effectiveness?
Q4: Do the AORE specifications produced by each approach have comparable quality?

Q1. Which activities are the main bottlenecks in terms of effort for each AORE approach?

Table 1 shows effort data that helps to answers this question. One commonality among all approaches is that the most time consuming activity was activity 2 (structure requirements specification). This makes sense as its goal is to gather the requirements of the identified concepts in activity 1 and build the AORE models in the specification document.

The composition specification sub-activity in this case was one of the most time-consuming activities of all (respectively 21%, 22%, 34% and 16% of the total effort spent in each approach). This is an interesting observation as as composition specification is an activity that is specific to AORE approaches and is not present in traditional RE approaches. The purpose of composition specification is to specify the rules about how the early aspects compose with the other concerns in the system. The expected benefit is that the separation of concerns is improved and thus change management and conflict analysis is achieved [4-6, 10]. In our case study we do not have data to support this claim. However, what we can highlight is that composition specification introduces a burden with respect to effort. Whether its benefits outweigh its overhead is something AORE techniques must demonstrate if they are to be deployed in day-to-day requirements engineering practices.

The data also shows that conflict analysis is also a time-consuming task. The level of detail of the conflict analysis varied between approaches because details of the interactions among early aspects varied due to variations in their granularity for each approach. For example, the goal-based AORE approach had very fine-grained crosscutting relationship specifications explaining why in absolute terms the effort in this approach (175 minutes) is much larger than the effort in the first two approaches whose compositions were more coarse-grained (54 minutes for Viewpoint-based AORE and 63 minutes for MDSOC).

However, despite the different levels of granularity, the important fact is that for all approaches conflict resolution and validation is a significant activity in terms of effort (30%, 21%, 29%, and 37%). An interesting point for future investigation would be to compare how conflict resolution activities compare considering AORE and traditional (non-AO) RE approaches. Metrics of effort could be collected to investigate how much more/less cumbersome is such conflict analysis in AORE and also how much more/less accurate AORE is (e.g., in terms of conflicts identified).

Q2: Is there any specific approach that significantly differs from the others in terms of effort?

The approach that significantly differs from the others in terms of effort is the Viewpoints-based AORE approach. The most significant differences are with respect to general activities 1 and 2 (identification and structuring) as most of these tasks were partially automated by the EA-Miner tool [19, 20]. We comment more on the reasons for this while answering the next question. Regarding the conflict analysis the effort spent was comparable to the MDSOC method as the specifications have a similar level of detail as commented in question 1 above.

Q3: What factors mostly contribute to differences in time-effectiveness?

Table 1 shows that activity 1 (identification of concepts) is significantly faster in the Viewpoints-based AORE approach due to the automation support provided by the EA-Miner tool. The tool utilizes corpus based natural language processing techniques (part-of-speech tagging, semantic tagging) and mining heuristics to automate the identification of viewpoints, early aspects and crosscutting relationships from a document written in natural language (in this case the health watcher use case document). The tool mines the

concepts and presents them to the user who can use some filtering and sorting features to decide what concepts to accept/discard. Therefore, this task was mostly automated by the tool and the engineer just had to look at the results and apply some filters to discard some irrelevant concepts. On the other hand, in the other approaches, engineers had to read the requirements document in order to identify the concepts and as the concepts are often spread through the document, the manual effort was significant.

Regarding the structuring of the requirements specification the tool also helps as it suggests grouping of requirements for the viewpoints and helps to automate the generation of an initial viewpoint-based AORE specification which we could extend later. In [21] we present data that shows how EA-Miner can save effort in the context of a different case study comparing 3 different systems in which we compared a manual versus an EA-miner based analysis for identification of concepts and producing an AORE specification based on the viewpoints approach.

On the other hand, tool support provided for conflict analysis (mainly by the AORA approach) does not necessarily show such a reduction in effort. This does not imply that the tool is not useful. The results could be attributed to the more detailed composition specifications in this approach which leads to fine-grained analysis of conflicts.

An interesting issue to explore in future case studies is to understand in-depth the benefits/drawbacks tools can bring to improve the quality of the produced AORE models and also the impact they have for time-effectiveness as we did for EA-Miner in [21]. Another factor that contributed to different results in time-effectiveness was the level of detail of the composition specifications that impacted the conflict analysis as discussed in question 1.

The next section compares the AORE approaches with respect to the quality of the specification models produced.

4.1.2 Quality of the Outcome Comparison

Being aware of the effort spent and what activities are cumbersome in the AORE approaches is important. However, another relevant question to assess is ***Q4: Do the AORE specifications produced by each approach have comparable quality?***

For example, even though Approach A might be faster than Approach B, but if the quality of the requirements specification of Approach A was inferior to Approach B this would explain the time results. In order to measure the quality of the outcome, we collected accuracy data of the identified concepts present in each requirements specification based on precision and recall as defined below:

- The precision for a technique Pt = (number of correct candidates identified by t) / (total number of candidates identified by t)**.**
- The recall for a technique Rt = (number of correct candidates identified by t) / (total known correct candidates).

Precision and recall metrics are general metrics used for measuring accuracy in several fields such as information retrieval and natural language processing. Similar to our purposes, aspect-mining code level approaches [22] also use these metrics to undertake comparisons of their mining capabilities in terms of the crosscutting concerns identified. Moreover, [23] collects precision and recall data to compare the capabilities of the OOPS tool for automating the generation of OO models against a manual analysis.

We use precision and recall to compare the different types of concerns identified based on our common naming scheme defined in Section 3. As the different AORE approaches rely on heterogeneous classifications of the concepts (e.g., early aspects, concerns, etc.) the naming scheme is important to map these differences onto the same schema thus enabling a uniform comparison among the approaches.

The researcher responsible for normalizing the results was the post-doc researcher who received the four different AORE requirements specifications. He mapped the identified concepts onto the three types of concepts defined in the common naming scheme (functional concern, functional early aspect and non-functional early aspect). This researcher was also responsible for determining if the identified concepts were correct or not (important for the precision and recall data). Every concept present in the documentation is considered a candidate and the post-doc researcher acted as an "oracle" to determine what was correct or not.

As can be observed in Table 3 each AORE approach identifies a different set of concerns and aspects. These differences highlight how each approach places a different emphasis on a certain type of concern (e.g., viewpoints-based approach does not consider functional early aspects). It is these differences that we wish to assess in order to determine which approach most closely identifies the complete list of concerns/aspects.

It is important to note that some approaches label certain concerns differently than others. For example, the MDSOC and AORA approaches identify a non-functional early aspect labelled Liability but in the original use-case document this concept is called Availability. These two concepts are clearly the same and both have the same effect on the system specification.

	Functional Concern	Non-Functional Early Aspect	Functional Early Aspect
Viewpoint-based AORE	Login, Register tables, Update Complaint, Register new employee, Update employee, Update health unit, Change employee Query information, Register complaint Information exchange **Correct candidates: 7 (underlined above)** **Total Number of candidates: 10** **Precision: 7/10= 70%** **Recall: 7/7= 100%**	Availability, Security, Performance, Concurrency, Persistence, Distribution, Error and exception handling, Compatibility, Usability, Legal Issues, Operational Environment. **Correct candidates: 10** **Total Number of candidates: 12** **Precision: 10/12= 83%** **Recall: 10/10= 100%**	None **Precision: 0%** **Recall: 0%**
MDSOC	Query Information, Complaint Specification, Register Tables, Register New Employee, Update Health Unit, Change Logged Employee, Update Employee, Update Complaint **Correct Candidates: 7** **Total Number of candidates: 8** **Precision: 7/8 = 88%** **Recall: 7/7 = 100%**	Integrity, Compatibility, Confidentiality, Liability, Performance, Usability, Security, Error Handling, Storage Medium **Correct Candidates: 7** **Total Number of candidates: 9** **Precision: 7/9 = 77%** **Recall: 7/10 = 70%**	Login **Correct Candidates: 1** **Total Candidates: 1** **P = 1/1 = 100%** **R = 1/2 = 50%**
AORA	Query Information, Complaint Specification, Register Tables, Update Complaint, Register New Employee, Update Employee, Update Health Unit, Change Logged Employee **Correct Candidates: 7** **Total Number of candidates: 8** **Precision: 7/8 = 88%** **Recall: 7/7 = 100%**	Usability, Liability, Performance, Security, Persistence. **Correct Candidates: 5** **Total Number of candidates: 5** **Precision: 5/5 = 100%** **Recall: 5/9 = 55%**	Login **Correct Candidates: 1** **Total Candidates: 1** **P = 1/1 = 100%** **R = 1/2 = 50%**
Goal Based with AORE AOV-graph	Specify Complaint, Update complaint, Login, Provide Information, Register Employee, Register Health Unit, Register Speciality, Register Disease, Register Symptom **Correct Candidates: 4** **Total Number of candidates: 9** **Precision: 4/9 = 44%** **Recall: 4/7 = 57%**	Persistence, Usability, Cryptography, Exception Handling **Correct Candidates: 4** **Total Number of candidates: 4** **Precision: 4/4 = 100%** **Recall: 4/10 = 40%**	Authentication **Correct Candidates: 1** **Total Candidates: 1** **P = 1/1 = 100%** **R = 1/2 = 50%**
Complete List	Register Tables, Update Complaint, Register New Employee, Register Complaint, Update Health Unit, Query Information, Update Employee **Total: 7**	Availability, Usability, Distribution, Security, Exception Handling, Persistence, Concurrency, Performance, Compatibility, Persistence **Total: 10**	Login, Authentication **Total: 2**

Table 3 – Precision and recall data of identified concepts in all approaches based on the common named scheme.

It is important that these differences are normalised for accurate comparisons to be made when considering precision and recall. However, in some cases entire concerns are categorised differently. For example, the MDSOC approach categorises Response Time and Throughput as two unique concerns whereas they should be consolidated into one concern labelled Performance.

Although the semantics of the requirements are the same (i.e. they contain the same concerns just organised and labelled differently) it is often these attributes that affect the comprehensibility of the documentation. Therefore, it is important that these differences are taken into account when performing any analysis related to precision and recall.

One of the clear results emerging from the analysis of the precision and recall data is that each approach generally performs better in identifying functional concerns over early aspects. This is explained by the fact that AORE is a novel technique raising difficulties even for experts in the field. In comparison the analysis of non-crosscutting concerns is a much more well understood phenomenon and is also simpler to perform due to such concerns being more localised and isolated in requirements documents.

In contrast there is much wider variation in the precision and recall results when considering early aspects. Generally these AORE approaches do have relatively good precision, in that the candidates they identify do in fact turn out to be early aspects. However, the majority of these approaches do have limitations when considering recall. These differences indicate that the AORE approaches are relatively accurate in identifying early aspects but are only able to identify a certain subset of these early aspects and not all of them.

The two concerns that seem to be the most problematic are Distribution and Concurrency. These two concerns are clearly crosscutting due to all functionality being distributed as well as, due to the persistent nature of the application, the necessity for concurrency control in all update/query operations to prevent inconsistencies. However, only one approach identifies these two concerns correctly as early aspects. This can be explained by the fact that concurrency is not explicitly mentioned but is instead an emergent property from the Performance and Persistence

concerns as the system must be able to support multiple users simultaneously (Performance) and could involve multiple read and write operations on stored data (Persistence). Concurrency emerges from the combination of these two requirements and hence is difficult to identify. The tool support used in the Viewpoints-based AORE approach is able to assist in the identification of such *derived* relationships more easily.

The results of the precision and recall advocate two future potential developments of AORE approaches. The first involves the further introduction of tool support, as identifying early aspects generally involves the simultaneous analysis of a number of concerns which cannot be easily done manually. The second development should involve the investigation of using multiple AORE approaches together, whereby the results are combined in union to allow the strengths of each approach to negate the drawbacks of the others.

5. Discussion

As discussed in Section 2, *explicit composition rules* are one of the main contributions of AORE. They are not utilized in non-AORE approaches. However, composition specification brings a considerable overhead in terms of effort compared to other common RE activities. Future studies should focus on investigating how much the benefits brought by the aspect modularity and composition, for example, improved change management, outweigh this overhead.

Some activities such as *identification and structuring* are also time-consuming and tool support is helpful to reduce this effort. Future studies could also focus on investigating conflict analysis tools in depth as this is also a time-consuming task.

The quality of the outcome of the different approaches is comparable in terms of accuracy of the specification produced measured by precision and recall data. Future studies could focus on other quality dimensions in terms of more fine-grained concepts such as the crosscutting relationships and the quality of the composition specifications.

The case study discussed here is a first stepping stone towards evaluation of AORE approaches. We are unaware of any similar case studies that investigated and compared different AORE approaches the way discussed here. One of the limitations of our study, regarding the generalization of its results, is related to conducting only one instance of the experiment. Since we have used as subjects people who are experts in each of the approaches and also because they were committed to the evaluation exercise, this single instance is a sound and reliable one.

Moreover, the definition of the common process scheme and common naming scheme also facilitate the realization of other case studies similar to ours. For example, one could conduct another case study to compare effort between AORE and non-AORE approaches using our case study as a guide. Therefore we think that despite its limitations our case study can be used as a first step in guiding future AORE evaluations.

6. Related work

Empirical works in requirements engineering (RE) are very varied covering several aspects of this discipline. For example, [24] describes an industrial study that evaluates a requirements engineering process maturity model as well as business improvements gained by suggested modifications on the requirements process of companies. [25] investigates the causes of faults in software systems that originate from errors in the requirements phase. An error abstraction process and a requirements error taxonomy are defined to help developers find errors in requirements specifications. An empirical study conducted with students shows that their approach helped to improve software quality and productivity of the subjects.

Regarding AORE approaches [4-7, 9, 10], we are not aware of any empirical studies conducted to assess quality aspects (e.g., effectiveness, productivity) in a comparative manner as well as investigate what benefits/drawbacks these approaches bring to the requirements practice. As AORE approaches are quite recent most works focus on demonstrating how the approaches work through the use of examples.

The problem of assessing different aspect-oriented approaches is mentioned in [22] with respect to code-level aspect-mining approaches. These researchers also had the problem of having to standardize their results to be able to compare the different techniques. To solve the problem they based their comparison on the concept of code-level crosscutting concern sorts to evaluate the quality of the different approaches. Even though these works focused on a different level of abstraction (i.e., code) they were very relevant for us as we had to address similar problems.

7. Conclusion

In this paper we have presented a comprehensive case study that is first yet key step towards improving AORE evaluation practice. We present a common framework (process and naming schemes) that enable uniform mapping of the discrepancies amongst existing AORE approaches in order to facilitate AORE evaluation exercises such as the comparison we conducted with four different techniques. This comparison yields interesting insights in that the

composition specification and conflict analysis activities are the most time consuming. Specifically, the results are intriguing as composition is the corner stone of AORE. Our study has not explored whether the benefits of such composition specifications more than compensate for this added effort not only from the perspective of requirements analysis but also from the viewpoint of deriving the system architecture and refining it towards detailed design and implementation. Such studies would be the focus of our future work as only then can we be convinced that aspect compositions address what Brookes [26] referred to as inherent complexity and do not introduce accidental complexity into the development process.

Acknowledgement: This work is supported by the AOSD-Europe and TAO Testbed projects. The authors also wish to thank Isabel Brito, Ricardo Ramos and Lyrene Fernandes for participating in the case study.

References

[1]I. Sommerville, et al., *Requirements Engineering - A Good Practice Guide*. 1997: Wiley.

[2]I. Sommerville, *Software Engineering*. 7 ed. 2004: Addision-Wesley. 784

[3]E. Baniassad, et al. *Theme: An Approach for Aspect-Oriented Analysis and Design*. in *International Conference on Software Engineering*. 2004. Edinburgh, Scotland, UK.

[4]A. Rashid, et al. *Modularisation and Composition of Aspectual Requirements*. in *2nd International Conference on Aspect Oriented Software Development (AOSD)*. 2003. Boston, USA: ACM.

[5]A. Rashid, et al. *Early Aspects: a Model for Aspect-Oriented Requirements Engineering*. in *International Conference on Requirements Engineering (RE)*. 2002. Essen, Germany: IEEE.

[6]J. Whittle, et al., *Scenario Modeling with Aspects*. IEE Proceedings - Software, 2004. **151**(4, Special Issue on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design): p. 157-172.

[7]Y. Yu, et al. *From Goals to Aspects: Discovering Aspects from Requirements Goal Models*. in *International Conference on Requirements Engineering*. 2004. Kyoto, Japan: IEEE Computer Socienty.

[8]G. Kiczales, et al., *Aspect-Oriented Programming*, in *11th Europeen Conf. Object-Oriented Programming*, M.A.a.S. Matsuoka, Editor. 1997. p. 220-242.

[9]E. Baniassad, et al. *Finding Aspects in Requirements with Theme/Doc*. in *Workshop on Early Aspects (held with AOSD 2004)*. 2004. Lancaster, UK.

[10]A. Moreira, et al. *Multi-Dimensional Separation of Concerns in Requirements Engineering*. in *Requirements Engineering Conference (RE 05)*. 2005. Paris, France.

[11]S.S. Jr, et al., *Modeling of Software Concerns in {Cosmos}*, in *Proc. 1st Int' Conf. on Aspect-Oriented Software Development {(AOSD-2002)}*, G. Kiczales, Editor. 2002. p. 127-133.

[12]R. Chitchyan, et al., *Report synthesizing state-of-the-art in aspect-oriented requirements engineering, architectures and design*. 2005, Lancaster University: Lancaster. p. 1-259

[13]V. Basili, et al., *Experimentation in Software Engineering*. IEEE Transactions on Software Engineering, 1986. **12**(7).

[14]B.A. Kitchenham, et al., *Preliminary guidelines for empirical research in software engineering*
IEEE Trans. Softw. Eng., 2002. **28**(8): p. 721-734.

[15]C. Wohlin, et al., *Experimentation in Software Engineering: An Introduction*, ed. V.R. Basili. 2000: Kluwer Academic Puslishers.

[16]E. Soeiro, et al. *An XML-Based Language for Specification and Composition of Aspectual Concerns*. in *ICEIS 2006*. 2006.

[17]S. Soares, et al., *Distribution and Persistence as Aspects*. Software: Practice & Experience, 2006. **36**(6).

[18]C. Potts. *Invented Requirements and imagine customers*. in *IEEE Requiremenst Engineering Conference (RE 95)*. 1995. York, UK.

[19]A. Sampaio, et al. *EA-Miner: A tool for automating aspect-oriented requirements identification.* in *20th IEEE/ACM International Conference on Automated Software Engineering (ASE2005)* 2005. Long Beach, California, USA.

[20]A. Sampaio, et al. *Mining Aspects in Requirements*. in *Early Aspects 2005: Aspect-Oriented Requirements Engineering and Architecture Design Workshop (held with AOSD 2005)*. 2005. Chicago, Illinois, USA.

[21]R. Chitchyan, et al. *Evaluating EA-Miner: Are Early Aspect Mining Techniques Effective?* in *TEAM workshop at ECOOP 2006*. 2006. Nantes, France.

[22]K. Mens, et al., *A Survey of Automated Code-Level Aspect Mining Techniques*. Transactions on Aspect-Oriented Software Development, Special Issue on Software Evolution, 2007.

[23]L. Mich, et al. *NL-OOPS: A Requirements Analysis tool based on Natural Language Processing*. in *3rd Int. Conf. On Data Mining*. 2002. Bologna, Italy.

[24]I. Sommerville, et al., *An empirical study of industrial requirements engineering process assessment and improvement.* ACM Trans. Softw. Eng. Methodol, 2005. **14**(1): p. 85-117.

[25]G.S. Walia, et al. *Requirement error abstraction and classification: an empirical study* in *ISESE'06*. 2006. Rio de Janeiro, Brazil: ACM.

[26]F.P. Brookes, *"No Silver Bullet - Essence and Accident" in The Mythical Man-Month*. 1995: Addison-Wesley. 177-203.