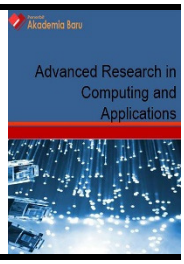
Penerbit Akademia Baru

Journal of Advanced Research in Computing and Applications

Journal homepage: www.akademiabaru.com/arca.html
ISSN: 2462-1927



Criteria and approach implications for requirements and design crosscutting concerns to support software evolution

Open Access

Jamaluddin Jasmis [1,*], Suhaimi Ibrahim [2], Shamsul Jamel Elias [3], Rosdiana Abd Razak [1], Wan Faezah Abbas [4]

[1] Faculty of Computer Science and Mathematics, UiTM (Melaka) Jasin Campus, 77300 Merlimau, Melaka, Malaysia.
[2] Advanced Informatics School, Level 5, Menara Razak, Universiti Teknologi Malaysia, Jalan Sultan Yahya Petra (Jalan Semarak), 54100 Kuala Lumpur, Malaysia.
[3] Faculty of Computer and Mathematical Sciences, UiTM, Kedah, 08400 Merbok, Kedah, Malaysia.
[4] Faculty of Science Computer and Mathematic, Universiti Teknologi MARA, 40450 Shah Alam, Malaysia.

ARTICLE INFO

Article history:
Received 4 February 2017
Received in revised form 28 February 2017
Accepted 1 March 2017
Available online 23 March 2017

Keywords:
Crosscutting concerns, Scattering, Tangling, AORE/AOSD, Software maintenance and evolution, IM-DeCRuD

ABSTRACT

Crosscutting concerns has gained special attention for software development and maintenance in software engineering. This awareness was resulted from the discovery of crosscutting behaviour that initially evolves from the process opening at implementation stage. Conceptually, crosscutting concerns are usually described in terms of scattering and tangling which involves mapping and intersection of software components throughout stages in development activities. Due to its orientation, it is inevitable to provide viable understanding in crosscutting concerns across the software lifecycle. AORE/AOSD targets at dealing with effective evolution process for crosscutting concerns at various phases in conjunction to industrial standard. Recent works are focusing on identification, modularization, composition and conflict analysis of crosscutting concerns solely at requirements level. However, there is significant research gap to appropriately specify crosscutting properties for functional and non-functional concerns at both requirements and design phases. Due to this absence, software engineers have no appropriate guidelines to attend to crosscutting concerns across development stages. In this paper, we aim to present several criteria published in literature scoping on crosscutting concerns at the requirements and design stages. In relation, we propose our approach called IM-DeCRuD to accommodate engineering tasks for better understanding and reasoning towards crosscutting concerns at the requirements and design stages. In this perspective, IM-DeCRuD is designed to be well suited with rapid changes of requirements for various sizes of software development as well as maintenance projects.

[*] *Corresponding author.*
E-mail address: jamaluddinjasmis@melaka.uitm.edu.my (Jamaluddin Jasmis)

1. Introduction

Concern is defined as "any matters of interest in a software system" which can be related to system functionalities or its properties [1]. It can be classified into functional (system's behavior or subsystems) and non-functional (system's properties). Rarely, these concerns act standalone as most of it will influence or constraint other concerns. These situations are known as *crosscutting* [2]. Crosscutting is usually described in terms of *scattering* and *tangling*. *Scattering* is the incidence where the source element (or a particular software component) influences (or map) to multiple target elements. *Tangling* in the other hand is referred to a case where several source elements intersect common target element [3] . For example, considering two concerns (or needs) of a particular information system, 'A' and 'B'. Given that a case that 'B' cannot be satisfied without affecting 'A' will give the meaning of concern 'B' *crosscuts* concern 'A'. 'B' might crosscut other concerns other than 'A', say, 'C'. So 'B' will also crosscut 'C' instead of 'A' and this situation is known as *scattering*. In other case, 'A' might be crosscut by 'D' other than 'B'. This situation is identified as *tangling*.

Crosscutting concerns has gained special attention for software development and maintenance in software engineering. This growing awareness was resulted from the discovery of crosscutting behavior that initially evolves from the process opening at implementation stage [4]. Due to its orientation, it is inevitable to provide viable understanding in crosscutting concerns across the software lifecycle. In relation, Aspect Oriented Requirement Engineering/Aspect Oriented Software Development (AORE/AOSD) has been introduced in recent years and gradually being accepted to be prominent technique in dealing with effective evolution process for crosscutting concerns. Hence, this new discovery has good research potential to be explored further in order to support engineers' tasks upon crosscutting concerns at various stages of software lifecycle, compliance to industrial technology standard [5].

Most of recent works are focusing on identification, modularization, composition and conflict analysis of crosscutting concerns solely at requirements level. It is due to its straightforwardness in dealing with high-level language in requirements documentations to specify concerns [6]. However, there is significant research gap to appropriately specify crosscutting properties for functional and non-functional concerns at both requirements and design phases. Due to this absence, software engineers have no appropriate guidelines to attend to crosscutting concerns across development stages [7]. In this paper, we aim to present several criteria published in literature scoping on crosscutting concerns at the requirements and design stages as potential benchmarks. In relation, we propose our approach called DeCRuD to accommodate engineering tasks for better understanding and reasoning towards crosscutting concerns at the requirements and design stages. In this perspective, DeCRuD is designed to be well suited with rapid changes of requirements for various sizes of software development as well as maintenance projects. This work is organized as follows. Section 2 shows related criteria related to requirements and design crosscutting concerns. Section 3 presents some related works. Section 4 promotes the results and discussion upon the evaluated approaches. Section 5 exhibits the proposed approach. Finally, Section 6 draws some conclusions and points to directions of future work.

2. Implication of criteria

These criteria were obtained and organized based on literature review. A brief explanation of each decisive factor can be found in the following sub sections as they will be used within the context of this research.

2.1. Concern identification

Concerns identification is dedicated to the identification process for discovering the decomposition units such as functional, non-functional concerns [6]. Concerns identifications can have variations such as *asymmetric* as well as *symmetric*. Asymmetric can be related to process of separation of what are functional and non-functional concerns whereas symmetric treats the decomposition units equally and regards everything is concern. Again, these concerns may be functional as well as non-functional concerns [6]. This research is directed to investigate the ability of the approaches to support asymmetric or symmetric identification of concerns.

2.2. Composability

Composability refers to the ability on how the crosscutting concerns and non-crosscutting concerns (standalone) are merged [6, 7]. Indeed, it is a natural way of reducing complexity and structuring enormous number of requirement decomposition units [8]. In this research, the composability of approaches under study is measured by investigating the ease of accomplishing that process.

2.3. Conflict resolution and decision support

Conflict resolution is ability for alleviating or eliminating sources of conflict among crosscutting concerns. Specifically, with the aid of decision support, it helps in setting trace-offs upon conflicting crosscutting concerns preceding by relationships composition as well as interactions validation and followed by weight assignments or priority setting processes [8]. Resolving conflicts is necessary as negatively contributing concerns may lead to undesirable influence on the entire composition and the system which result in poor architecture [7]. As for the approaches under study, identifications are done to see whether this criteria is supported or else.

2.4. Traceability across software lifecycle

As mentioned earlier, traceability is referred to the ability to trace within and between software components. It is crucial for quality software in term of its understandability, maintainability and manageability [8]. Specifically, this criterion is divided into requirements traceability and traceability between components across development stages [8]. Adhering to the focus of this study, benchmarking is made upon the ability of crosscutting as well as standalone concerns being traced among requirements and design stages.

2.5. Support for mapping

Approach is measured in term of their ability to progress to later stage which is referred to formal design components [8]. Indication is made if the approach provides support for decision on mapping especially for crosscutting concerns. It is crucial as it will facilitate efficient solution choice pertaining to recording, communication and management activities [8].

2.6. Evolvability

Evolvability refers to the ability to change or removal/addition of requirements [8] and propagate its effects to design. Any effect upon changes situations practically, should be made localized and

easy to implement to ensure those components in the early development stages up to date with the changing requirements. This is stressfully needed for understandability and management support [8]. In this research, the evolvability of approach is measured by investigating the ease of accomplishing such process.

2.7. Conflict resolution and decision support

Scalability is referred to the ability of approach to be suited well for small and large projects. This is stressfully important to support growing projects management [8].

3. Related works

AORE/AOSD has greatly been explored in few years' back that continuously focuses on providing software engineers with method and tool supports. It is aimed at identification, modularization, composition and conflict analysis upon crosscutting concerns across earlier phases of development lifecycle.

Rosenhainer suggested a semi-automated AORE method called Dealing Separately with Crosscutting Concerns (DISCern) [9, 10] to identify and specify functional and non-functional crosscutting concerns in pre-existing requirements documentation supported by information retrieval techniques. However, this method does not specify how crosscutting concerns would be formally represented in both requirements and design phases.

Identification of Crosscutting Concerns with UML is an AORE approach which has been proposed by Brito [11] and Araújo [12] with the aim of handling the separation of crosscutting concerns at the phase of requirements. In this approach, UML model is used as a basic notation to specify crosscutting functional and non-functional crosscutting concerns. Similarly, Xiaomei *et al.* [13, 14] present an AOSD approach called Non-Functional Requirements/Aspectual Use Case Driven (NFR/AUC). NFR/AUC is an extension approach to accommodate seamless transition of crosscutting concerns between requirements and design phases. This composition task can be achieved by identifying finer grain of non-functional crosscutting concerns through NFR Framework[2] and incorporated the framework into use case diagram (functional concerns) via association points. Unfortunately, these methods require highly skilled engineers to accomplish and lack of tool support, thus it is applicable to only small-scaled projects. Furthermore, these methods do not specify how to deal with maintenance tasks.

A semi-automated AORE approach introduced by Sampaio *et al.* [15] that utilizes Corpus-based Natural Language Processing (NLP) for early-phase crosscutting concern identification. Early Aspect Identification Method (Early-AIM) enables requirement engineers to quickly identify and to form a structured crosscut-oriented model as a result of automatic mining process upon structured or unstructured sources of requirements (e.g. documents, interviews, natural language descriptions of the system), using EA-MINER tool [16, 17]. Besides to its limitation which is like DISCern, Early-AIM fails neither to support conflict analysis nor to specify non-functional-functional crosscutting concerns.

Another semi-automated AORE approach has been proposed to identify and specify crosscutting concerns by applying "extensible" XML-based composition rule for their composition. Model is conceptualized by employing XML-based composition language using Aspectual Requirements

[2] *NFR Framework* is a mechanism that decomposes softgoals of non-functional concerns into principles and operational sub-goals in form of tree structure

Composition and Decision Support Tool (ARCaDe) [18, 19]. It possesses similar limitation as DISCern, besides fails to specify on its applicability on software project size.

Somehow like NFR/AUC, Aspectual Unified Software Development Process/Non-Functional Requirements Driven Approach (AUSDP/NFR) was proposed to provide new relationship mechanism to represent crosscutting relationship between NFR Framework and use cases which is applicable at requirements and design phases. However, it exposes to limitations like Identification of Crosscutting Concerns with UML and NFR/AUC. Likewise, it does not provide a seamless transformation notation for crosscutting concerns between requirements and design phases [20], particularly the detailed design.

4. Evaluation criteria

We formulate the evaluation criteria that were based on [6, 8]. The capability level of approach is analyzed in meeting some evaluation criteria such as Identification[3], Composability, Scalability (suitability on various sizes of systems), Traceability across Software Lifecycle, Support for Mapping, Evolvability and Conflict Resolution & Decision Support. We use suitable indicators for the capability level - *Yes* (fulfill), *No* (not fulfill), *N/S* (Not Specified) and **–** (Not Applicable).

Table 1 depicts evaluation criteria used to compare selected approaches in the direction to develop an improvised approach to support requirements and design crosscutting process.

Early-AIM approach which leads to asymmetric requirements identification has several drawbacks to be noted. Firstly, is referring to lack of support on semantic composition upon non-functional-functional crosscutting relationships. Secondly, no such mechanism that can identify any conflicts arises and decision support for solution. Early-AIM also does not specify significant links as well as mapping between requirements and design artifacts.

Somehow like Early-AIM, AUSDP/NFR which categorized under asymmetric requirements fails to support composability. Even it provides better support upon conflict resolution and traceability; mapping, evolvability and scalability specifications are aborted.

Table1
Summary of result on the comparison analysis of the selected approaches

Appr. / Criteria	DISC-ERN	Id. of Cross. Concerns with UML / AUC/NFR	Early-AIM	ARCa -DE	AUSDP/NFR
Identification					
Symetric	-	-	-	Yes	-
Asymmetric	Yes	Yes	Yes	-	Yes
Composability	Yes	No	No	No	No
Conflict Resolution & Decision Support	Yes	Yes	No	Yes	Yes
Traceability across SW Lifecycle	N/S	Yes	No	Yes	Yes
Support for Mapping	N/S	Yes	N/S	Yes	N/S
Evolvability	No	N/S	Yes	Yes	N/S
Scalability	Yes	N/S	Yes	N/S	N/S

[3] Types of requirements identified. *Symmetric* (any of functional and non-functional) while *Asymmetric* (both type)

It seems that combination of IR-based, Identification of Cross. Concerns with UML / AUC/NFR and ARCADE approaches score the best based on the evaluation criteria. It is meant as these three approaches complement each other in some criteria that are lacking. As such, it is believed that this combination will yield a better direction to develop the proposed approach as such.

5. IM-DeCRuD approach

Our research adopts and integrates certain techniques from the existing approaches as in [21] to develop an approach called Identification, Modularization, Design Composition Rule and Conflict Dissolution (IM-DeCRuD) that supports engineers to manage crosscutting concerns at requirements and design phases. IM-DeCRuD is tailored to suit with rapid changes of requirements for various sizes of software development as well as maintenance projects. As depicted in Figure 1, IM-DeCRuD can be divided into three main tasks and will be described further in the following subsections.

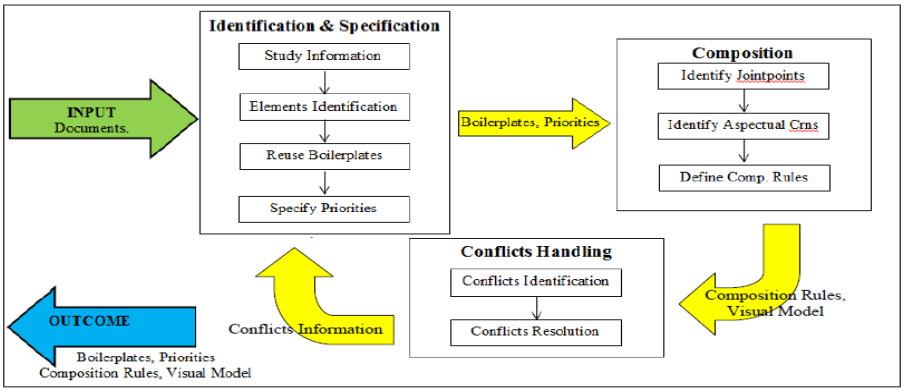


Fig. 1. The proposed IM-DeCRuD approach

5.1. Identification and specification

Identification and Specification is the starting point of the approach which involves in compiling and reviewing high level requirements documentations, followed by extraction process upon available nouns, verbs and system's properties from each requirement for viewpoints, FURs (functional concerns) and Product-Oriented NFURs (non-functional concerns) respectively by the domain experts. Next, process of compiling, organizing and recording those requirements components from multiple sources is performed by applying specific-purpose boilerplates. This subtask is fulfilled by specifying priorities of the identified NFURs as per determined by the stakeholders, with the value taken as 'High', or 'Low' for conflicts identification purposes that will be carried out later in another subtask.

5.2. Composition

Composition carries out with the possibility to weave together those identified requirements components. For this specific purpose, our proposed composition rules in which the structure is governed by XML schema with its operators are adopted from LOTOS (Language of Temporal Ordering Specifications) [22] is applied. This composition rule acts as a meta-language to accommodate formation of graphical notation as an alternative view for standard software high level design since the later seems to have limitation for the purpose to specify crosscutting concerns [1, 2, 11].

5.3. Conflict handling

Finally, Conflicts Handling deals with the identification task towards conflicting crosscutting concerns (NFURs) which are similar or falls under shared category besides having common priorities and conflicting quantification measurement that constraint common conventional concern (FUR). Next, these conflicting crosscutting concerns issues will be communicated and negotiated (dissolution) with the stakeholders.

6. Conclusion

We have presented several criteria from literature which guided us in developing DeCRUD approach to support crosscutting concerns evolution process between requirements and design phases. This approach is tailored to accommodate bi-directionally links between main tasks for effective evolution process for various sizes of software development and maintenance projects. For future works, we will develop a tool and test it against suitable case studies.

References

[1] Ali, Busyairah Syd, and Zarinah Mohd Kasirun. "Developing tool for crosscutting concern identification using nlp." In *Information Technology, 2008. ITSim 2008. International Symposium on*, vol. 3, pp. 1-8. IEEE, 2008.

[2] Ali, Busyairah Syd, and Zarinah Mohd Kasirun. "A Review on Approaches for Identifying Crosscutting Concerns." In *Advanced Computer Theory and Engineering, 2008. ICACTE'08. International Conference on*, pp. 855-859. IEEE, 2008.

[3] Conejero, José M., and Juan Hernández. "Analysis of crosscutting features in software product lines." In *Proceedings of the 13th international workshop on Early Aspects*, pp. 3-10. ACM, 2008.

[4] S. S. Geórgia Sousa, Paulo Borba and Jaelson Castro, "Separation of Crosscutting Concerns from Requirements to Design: Adapting an Use Case Driven Approach," in *Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design Workshop of the 3rd International Conference on Aspect-Oriented Software Development,* Lancaster, UK., 2004.

[5] Hannemann, Jan, Ruzanna Chitchyan, and Awais Rashid. "Analysis of aspect-oriented software." In *European Conference on Object-Oriented Programming*, pp. 154-164. Springer Berlin Heidelberg, 2004.

[6] Sampaio, Americo, Phil Greenwood, Alessandro F. Garcia, and Awais Rashid. "A comparative study of aspect-oriented requirements engineering approaches." In *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, pp. 166-175. IEEE, 2007.

[7] Khan, Safoora Shakil, and Muhammad Jaffar-ur-Rehman. "A survey on early separation of concerns." In *Software Engineering Conference, 2005. APSEC'05. 12th Asia-Pacific*, pp. 7-pp. IEEE, 2005.

[8] Chitchyan, Ruzanna, Awais Rashid, and Peter Sawyer. "Comparing Requirements Engineering Approaches for Handling Crosscutting Concerns," in *8th Workshop on Requirements Engineering*, 2005.

[9] Rosenhainer, Lars. "The DISCERN method: dealing separately with crosscutting concerns." In *Early Aspects Workshop at OOPSLA-2005*. 2005.

[10] Rosenhainer, Lars. "A Method for Handling Requirements-Level Crosscutting Concerns." *Softwaretechnik-Trends* 26, no. 1 (2006).

[11] I. Brito, "Aspect-Oriented Requirements Engineering," in *7th International Conference on Unified Modelling Language (UML)* Lisbon, Portugal, 2004.

[12] J. Araújo, A. Moreira, I. Brito, and A. Rashid, "Aspect-Oriented Requirements with UML," in *Workshop on Aspect-Oriented Modelling with UML*, 2002.

[13] Liu, Xiaomei, Shulin Liu, and Xiaojuan Zheng. "Adapting the NFR framework to aspectual use-case driven approach." In *Software Engineering Research, Management and Applications, 2009. SERA'09. 7th ACIS International Conference on*, pp. 209-214. IEEE, 2009.

[14] Zheng, Xiaojuan, Xiaomei Liu, and Shulin Liu. "Use case and non-functional scenario template-based approach to identify aspects." In *Computer Engineering and Applications (ICCEA), 2010 Second International Conference on*, vol. 2, pp. 89-93. IEEE, 2010.

[15] Sampaio, Américo, Awais Rashid, and Paul Rayson. "Early-aim: An approach for identifying aspects in requirements." In *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, pp. 487-488. IEEE, 2005.

[16] Sampaio, Américo, Ruzanna Chitchyan, Awais Rashid, and Paul Rayson. "EA-Miner: a tool for automating aspect-oriented requirements identification." In *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, pp. 352-355. ACM, 2005.

[17] C. Ruzanna, S. Américo, R. Awais, and R. Paul, "A Tool Suite for Aspect-Oriented Requirements Engineering," in *Proceedings of the 2006 International Workshop on Early Aspects at ICSE* Shanghai, China: ACM, 2006.

[18] Rashid, Awais, Ana Moreira, and João Araújo. "Modularisation and composition of aspectual requirements." In *Proceedings of the 2nd international conference on Aspect-oriented software development*, pp. 11-20. ACM, 2003.

[19] Rashid, Awais, Peter Sawyer, Ana Moreira, and João Araújo. "Early aspects: A model for aspect-oriented requirements engineering." In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, pp. 199-202. IEEE, 2002.

[20] Meier, Silvio, Tobias Reinhard, Reinhard Stoiber, and Martin Glinz. "Modeling and evolving crosscutting concerns in ADORA." In *Aspect-Oriented Requirements Engineering and Architecture Design, 2007. Early Aspects at ICSE: Workshops in*, pp. 3-3. IEEE, 2007.

[21] J. Jasmis, S. Ibrahim, and R. B. Anom, "AORE/AOSD Approaches For Requirements and Design Crosscutting: A Review," in *International Conference on Software and Information Engineering (ICSIE 2011)*, in press.

[22] Marsan, Marco Ajmone, Andrea Bianco, Luigi Ciminiera, Riccardo Sisto, and Adriano Valenzano. "A LOTOS extension for the performance analysis of distributed systems." *IEEE/ACM Transactions on Networking (TON)* 2, no. 2 (1994): 151-165.