Arab J Sci Eng (2015) 40:475–486
DOI 10.1007/s13369-014-1454-3

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

Resolving Aspect Dependencies for Composition of Aspects

K. Santhi · G. Zayaraz · V. Vijayalakshmi

Received: 24 May 2014 / Accepted: 30 September 2014 / Published online: 26 November 2014

Abstract A new modularization technique is used in Aspect-oriented software development for the separation of widely used functions such as logging, caching, synchronization, and exception handling from the core business logic functions. Aspects are identified using the mathematical modeling tool, Colored Petri nets. During the software development process, dependencies may arise as a result of using operators such as Before, After, Around, and Replace; such dependencies are consumed by our framework. Using the specification of aspects, we generate a composition rule for every match point, which directs the composition process at the initial requirements phase of software development. The proposed FTS approach, incorporating a feedback edge set, topological ordering, and second valid ordering, is efficient in resolving conflicts and dependencies among the aspects. To analyze the second valid ordering, grey relational analysis is used to rank the aspects, while analysis of variance method is used for the verification thereof. The proposed approach is illustrated by a case study.

Keywords Colored Petri net · Feedback edge set · Grey relational analysis · Match point · Second valid ordering · Topological ordering

K. Santhi (✉) · G. Zayaraz
Department of CSE, Pondicherry Engineering College, Puducherry India
e-mail: santhikrishnan@pec.edu

G. Zayaraz
e-mail: gzayaraz@pec.edu

V. Vijayalakshmi
Department of ECE, Pondicherry Engineering College, Puducherry India
e-mail: vvijizai@pec.edu

1 Introduction

The software development life cycle (SDLC) gives a basic structure for developing software products. In software development, coupling is defined by the number of dependencies between two subsystems, while cohesion is calculated as the number of dependencies within a subsystem [1–4]. Scattering and tangling behaviors arise as a result of crosscutting concerns in software development; these behaviors are encapsulated in separate modules and referred to as aspects [3–5].

Aspect-oriented requirements engineering, otherwise known as "early aspects," focuses on the detection, encapsulation, description, and composition of crosscutting properties at the requirement level itself [6]. In the past few years, there has been mounting interest in propagating the aspect model. In the core module, aspects are separated from one another to reduce dependencies between modules, so as to automatically increase the modularity of the system; hence, a high level of cohesion and low level of coupling are achieved [4,5]. To improve the quality of the software system, it should have high maintainability, which is achieved only if the system is modifiable, extensible, and reusable [7–10]. Some interactions between modules may generate the anticipated behavior, whereas others may cause unanticipated behavior of the system. Thus, it is feasible to identify the interactions and possible inconsistencies in the software development process, if possible, at the requirements phase itself [8,11]. Resolving such conflicts at the requirements stage is more efficient, faster, cheaper, and more desirable than carrying out essential code later on-the-fly [11,12].

Therefore, it is necessary to include aspects as essential modeling primitives at the initial requirements phase of the software development engineering process. The objectives of our framework are twofold:

(a) Identifying crosscutting concerns are achieved using Colored Petri nets (CPNs).
(b) Detecting and resolving the conflicting concerns early, before the architecture has been designed, using the FTS (feedback edge set, topological ordering, and second valid ordering) approach.

The rest of the paper is organized as follows: Sect. 2 briefly describes an existing aspect-oriented approaches, CPNs, topological ordering, and second valid ordering. Section 3 describes related works in aspect-oriented requirements engineering, while Sect. 4 summarizes the FTS approach. Section 5 shows how the main idea can be implemented in a case study, and finally, Sect. 6 draws some conclusions and presents suggestions for future work.

2 Background

Throughout the software development process, the Aspect Oriented Software Engineering Group has aimed to develop systematic means for the detection, encapsulation, description, and composition of crosscutting concerns, otherwise known as aspects [13–15]. The approach in aspect-oriented programming (AOP) is supplemented by analogous techniques previously used with regard to realizing crosscutting concerns, which pervade throughout all phases of the SDLC. The inspiration for AOP approaches is to modularize crosscutting concerns. In a module, there is the chance that concerns overlap each other; if this is the case, it is referred to as tangling. Likewise, there is a chance that concerns can isolate a module from other modules, which is referred to as scattering. Even though scattering and tangling are different concepts, they can exist together in modules. The execution of a concern is scattered over multiple modules and can affect the execution of multiple modules as shown in [4,5]. Hence, the goal of our work is to develop a framework for aspect-oriented requirements engineering (AORE) at the requirement level itself, which supports the detection and separation of functional and non-functional concerns and the composition specification of match points, in which one or more aspects are applied, and which is used to detect conflicts.

AORE methods are used to address the composability and subsequent analysis of crosscutting concerns during the requirements engineering (RE) process [16–19]. However, performing aspect composition in AORE is based on syntactic references to requirements in the base; for instance, direct references to requirement identifiers in the system. Based on a simplistic view, the AORE process is divided into two main parts:

(1) Crosscutting concerns: The process first handles non-functional requirements by identifying which concerns crosscut other concerns and then performs a traditional specification of the functional requirements of the software system by identifying which concerns are crosscut by other concerns.
(2) Composed requirements: The process starts by composing functional requirements with aspects and then identifies and resolves conflicts that may arise from the composition process. This is accomplished through the use of the following operators.
Overlapping: The requirements of the aspect modify the functional requirements they crosscut. Overriding: The aspect requirements superpose the functional requirements they transverse. Wrapping: The aspect requirements are "encapsulated" with the functional requirements they transverse.

Figure 1 gives an overview of the model for aspect-oriented requirements analysis. Many design problems, such as errors and flaws, can be identified and resolved in the requirements and design phases rather than in the implementation phase [2,21]. This process can reduce the difficulties in the implementation and test phases of the software cycle. CPNs describe the states of the system and the events (transitions) that cause changes in these states. By performing simulations in a CPN model, it is possible to inspect different states and explore the behavior of the system as in [1,20,22], which are very useful for debugging and investigating the system design for simulating CPN tools. Topological ordering is possible only for DAGs [23]; if a cycle occurs, it is not possible to perform this ordering. To overcome this problem, the feedback edge set method can be used to remove any cycles in the graph after which topological ordering can be performed. While performing topological ordering, conflicts may occur, causing there to be more than one possible ordering. In this situation, to overcome the conflict a second valid ordering method is used. This method swaps the order of the conflicting vertices by considering their priority and gives a valid and unique topological ordering. Second valid ordering is needed whenever the topological ordering is not unique. In this case, it is always possible to swap two consecutive vertices that do not have an edge between them.

3 Related Work

Several methods, such as those applying formal languages, syntactical analysis, and composition filters, among others, are concerned with aspect interactions and their dependencies. These methods are used to focus on the analysis and verification of aspects as in [24,25] which provides the key for aspect interaction. Formal language and syntactical analysis are used to detect interactions between aspects and to resolve conflicts.

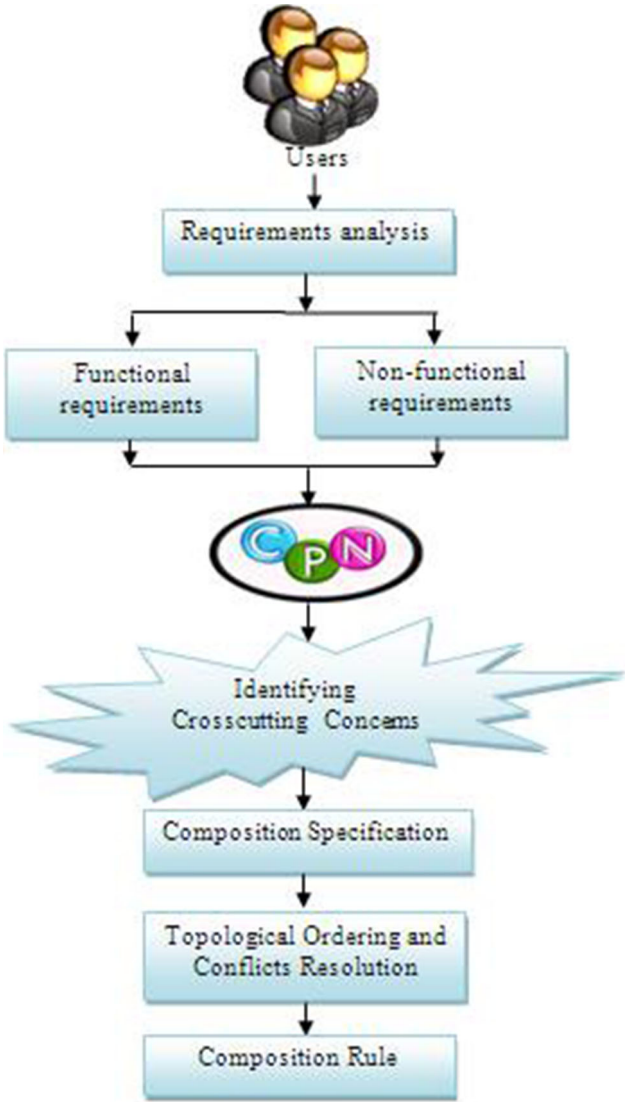


Fig. 1 AORE analysis

Mehner et al. [6] focused on crosscutting concerns that caused conflicts to occur between them. Their method used composition filters to deal with the difficulty of handling crosscutting concerns and their interactions. During the analysis phase, a number of techniques and results have been considered to deal with conflicting conditions, such as [2,4,6,13,16]. Estimated a method for analyzing and detecting interactions between functional and non-functional concerns, crosscutting each other. Using a graph transformation tool, the performance of detecting conflicts and their dependencies is analyzed through the activities used to process the use cases.

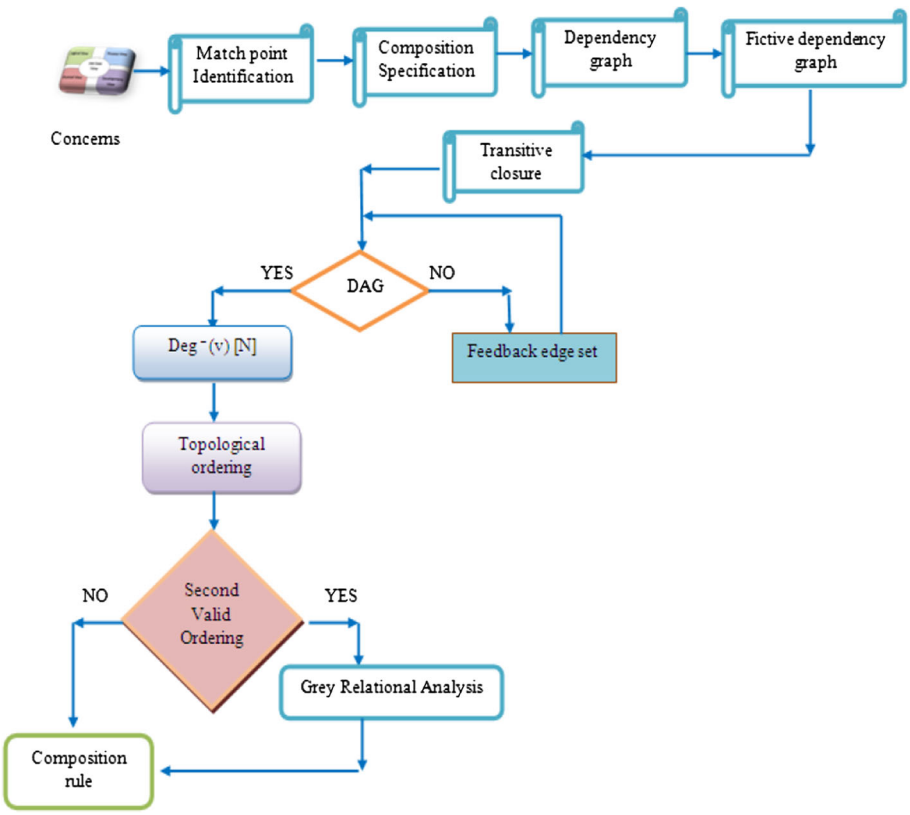
Shen et al. [8] provided a method applied in the requirements phase to deal with non-functional crosscutting concerns. They used UML models to identify the functional and non-functional concerns. If any conflict occurs between these, it is resolved based on the stakeholder's opinions. If a negative contribution of crosscutting concerns occurs, affecting a single concern, the authors recommend trade-off analysis based on the stakeholder's opinions.

Brito and Moreira [10] explained the constitution of crosscutting concerns with functional concerns. In their model, match point identification is used to generate a composition rule and if conflict occurs at the match point, dominant crosscutting concerns are used to resolve it. A composition rule is defined for a match point; to resolve the conflicts, there is a need to identify the dominant crosscutting concerns.

Rachid et al. [11] suggested a generic aspect-oriented requirements model based on viewpoints and the eXtensible Markup Language (XML) to separate the aspectual requirements and non-aspectual requirements. Aspectual requirements assist in the generation of composition rules; when a conflict occurs, trade-off analysis is performed in the system based on the stakeholders' opinions. A contribution matrix and attribute weights are used to resolve the conflicting aspects.

Sofian et al. [26] explained the requirements negotiation process used to reach agreement among stakeholders regarding the requirements of the software to be developed. The process is an abstract representation of activities involved in performing requirements negotiation, in which the outcomes are agreed requirements defining the functionality and non-functionality of the software. Misunderstanding or misidentification of the functionality of software attributes may result in decreased quality of the software application [3,12,27]. Additionally, the negotiation process may identify unresolved issues that must be considered, since these may dramatically affect the functional and non-functional attributes of the software. Identification of requirements entails the involvement of numerous heterogeneous stakeholders with different roles, responsibilities, and priorities; hence, conflicts are likely to occur in some requirements [28]. As the group learns aspect-oriented programming and begins including more aspects into projects, there is the possibility that several aspects could be added to the same pointcut or match point. Aspects should be ordered; for instance, an authorization aspect must be executed before a caching aspect. Even if some aspects do not lend themselves to ordering, they should be applied in a predictable order, otherwise, an aspect that functions correctly today, may not do so the next time simply because the aspects were applied in a totally different order. Since some aspects may be conflicting, they cannot be added to the same pointcut without being ordered. For instance, it does not make sense to make an object persistent using two different aspects: one causing persistence to the database, the other to the registry. Moreover, some aspects require other aspects to be applied; for instance, an aspect changing the mouse pointer to an hourglass requires the method to execute asynchronously; otherwise, the cursor shape would never be restructured. Thus, an effective

Fig. 2 FTS approach



requirements negotiation process is vital for stakeholders to discuss and resolve inconsistencies and conflicts [28,31] to view the software to be developed from the same perspective.

Boubendir and Chaoui [4,5] proposed a method to help the user identify interactions between aspects and resolve the conflicts between them at the requirements analysis phase. Operators such as Before, After, Around, and Replace is used to generate various dependencies, which are used in this framework. Then, a composition rule is generated using the Hamiltonian path. If a Hamiltonian path is not available in the graph, the longest path is used to resolve conflicts among the concerns. In this paper, we propose a method that extends the work done by [4,5], which enables the user to detect interactions between crosscutting concerns and identify and resolve the conflicts between them in the requirements phase. The extensions include reducing the time complexity of computing the composition rule by using an efficient algorithm such as topological ordering, second valid ordering, or grey relational analysis. The time complexity of the proposed algorithm is cubic, as opposed to the algorithm in [4,5], which is NP-complete.

4 FTS Approach

A software system comprises a number of concerns, each of which consists of one or more functional requirements, non-functional requirements, or a combination of these. We propose that FTS approach for analyzing interactions in a match point is shown in Fig. 2.

Operators such as Before, After, Around, and Replace can generate dependencies, which are used to specify the composition of aspects. By considering this, the dependency graph, fictive dependencies (artificial dependencies) if any, and transitive closure of the graphs can also be generated. This allows composition rules to be generated for every match point, which can be used to guide the process of composition in the requirements phase itself [29]. If the resulting graph is cyclic, the feedback edge set method is applied to acquire a DAG so that topological ordering can be performed. Topological ordering is possible only for DAGs; if a cycle occurs it is not possible to perform this ordering. To overcome this problem, the feedback edge set method can be used to remove any cycles in the graph after which topological ordering can be performed. While performing topological ordering, conflicts may occur, causing there to be more than one possible ordering. In this situation, to overcome the conflict a second valid ordering method is used. This method swaps the order of the conflicting vertices by considering their priority and gives a valid and unique topological ordering. Second valid ordering is needed whenever the topological ordering is not unique. In the second valid ordering, grey relational analysis [30–32] is used to solve multiple attribute decision making by the groups of stakeholders.

FTS Algorithm

```
1.  MatchPointIdentify (Transitions)
2.   {
3.     while ( ! EOF (concern)) do
4.      Perform Requirements Analysis
5.      For each requirement
6.      {
7.      Construct Requirement Net (RN) = (PN, LE)
         //PN pertinent
8.     Construct (LE= (O1, O2, O3……., On))
     //LE logical entity
9.       }
10.     Construct Concern Net & Execution Order
11.      if (Dependency && Relations) then
12.      Specify Tokens and Transition T
13.      else
14.      Declare No crosscutting exists in T
15.      end if
16.     end while
17.  while(!EOF(Transition))do
18.  {
        // Deg - (T) is functional and non-functional
          concern
19.     if (Deg - (Candidate Aspects (T) > =1))then
20.     T is designated as Match Point
         // Generate Composition Specification
21.      for each Candidate Aspect
22.       if  (CA(i) > T) then T→ CA(i)
          // Before Operator
23.       else if (CA(i) < T)then T ←CA(i)
         // After Operator
24.       else if (CA(i)|| T) then T =>CA(i)
         // Around Operator
25.       else CA(i) ⋯⋗ T
         // Modify Operator
26.      end if
27.     end for
28.    else
29.    Tis designated as No Match Point
30.   end if
31.  }
32. end while
33. Generate Initial Dependency, Fictive & Transitive
    Closure Graph
34. while! (Is DAG ())do
35.    {
36.     Perform Feedback Edge Set
        // Remove an edge in the graph
37.    }
38.  Compute Deg - (v)[N]
    //Find the in-degree of the nodes
39. Perform Topological Ordering
40. while Second Valid Ordering needed
41.     do Grey relational analysis
42. end while
43. Generate Composition Rule
44. }
45. end while
```

The working principle of the proposed FTS algorithm is that each and every requirement is considered an independent Petri net. For each requirement, we define a requirement net, and for each requirement net logical entities are found. A requirement net is defined by the 2-tuple, RN = (PN, LE), where PN is the Petri net, $|P| = 2$, $|T| = 1$, $|F| = 2$, and LE = (O1, O2, . . . , On) is a set of logical entities. The execution order is specified for each concern thereby constituting a concern net, which is defined by the 2-tuple, CN = (SoR, SoE), where SoR = (RN1, RN2, …, RNn) ($n > 0$), is a finite set of requirement nets, and SoE = (EO1, EO2, …, EOn) ($n > 0$), is a finite set of execution orders. An execution order is a sequence of requirement nets, EO = (RN1, RN2, …, RNn).

Requirement nets and the execution order are needed to create a concern net, and the execution of which requires that the correct token is placed in the first space in the concern net. If there are not enough tokens in the first space, the concern net cannot be executed correctly in the final Petri net model and it is impossible to identify the aspects of the system.

We identify the dependencies, restrictions, and relationships among the requirement and concern nets. A new space is created temporarily if a dependency occurs between two requirement nets and a dependency token is placed in it. Complete execution of the system is achieved through the names of the token and requirement nets. After specifying the transitions of each concern net with more entry points, if the values of the entrance tokens are distinct, both the entrance and transition tokens are taken as token1, token2. Next we determine the logical entities related to the requirement nets. The tangling problem arises in logical entities that belong to two requirement nets with the 2-tuple (token1, token2). Once crosscutting concerns have been identified, we calculate the in degree of the transition. An in degree greater than one implies that both functional and non-functional requirements of the system exist, resulting in a match point. The aspect takes an action at this match point at the appropriate time such as Before, After, Around, and Modify.

By considering this, the dependency graph, fictive dependencies (artificial dependencies) if any, and transitive closure of the graphs can also be generated. If the resulting graph is cyclic, the feedback edge set method is applied to acquire a DAG so that topological ordering can be performed by calculating the in degrees of aspects in the graph. While performing topological ordering, conflicts may occur, causing there to be more than one possible ordering. In this situation, to overcome the conflict, a second valid ordering method is used, which in turn uses grey relational analysis to resolve conflicts.

Table 1 Topological ordering process

V	In-degree							
P	3	2	1	0	–	–	–	–
A1	5	4	3	2	1	0	–	–
A2	0	–	–	–	–	–	–	–
A3	4	3	2	1	0	–	–	–
A4	2	1	0	–	–	–	–	–
A5	5	4	3	2	1	0	0	–
A6	0	0	–	–	–	–	–	–
Enqueue	A2 A6	A6	A4	P	A3	A1 A5	A5	Nil
Dequeue	A2	A6	A4	P	A3	A1	A5	Nil

Consider the example discussed in [4,5], where the candidate's aspects A1, A2, A3, A4, A5 and A6 affect match point (join point) P. A match point is where the behavior of functional and non-functional concerns join [6]. Suppose that aspect A1 overlaps before the match point, aspect A2 overlaps after the match point, aspect A3 wraps the match point, aspect A4 is a substitute for the match point, aspect A5 overlaps before the match point, and aspect A6 overlaps after the match point. For the Before operator ($\rightarrow$), the match point is satisfied only after the aspect's satisfaction. Thus, match point P is satisfied by the satisfaction of aspects A1 and A5; hence, P $\rightarrow$ A1 and P $\rightarrow$ A5.

For the After operator ($\leftarrow$), the match point must be satisfied before satisfying the aspects and the behavior of aspects A2 and A6 must be attached after P; hence, P $\leftarrow$ A2 and P $\leftarrow$ A6. For the Around operator ($\Rightarrow$), the behavior of aspect A3 must be satisfied in parallel with the behavior of join point P; P $\Rightarrow$ A3. For the Replace operator ($\dashrightarrow$), the operator substitutes the behavior of A4 for the behavior of P; hence, A4 $\dashrightarrow$ P. To locate the fictive dependencies, Around and Replace operators are used. For operator Around, the behavior of aspect A3 must be satisfied in parallel with the behavior of join point P; hence, A3 $\rightarrow$ A1 and A3 $\rightarrow$ A5 are identified.

For operator Replace, aspect A4 modifies the behavior of join point P. Therefore, it is concluded that, there exists a concrete probability that all aspects depending on join point P are dependent on aspect A4. Fictive dependencies A6 $\rightarrow$ A4 and A2 $\rightarrow$ A4 are identified. For aspects X, Y, and Z, if X depends on Y and Y depends on Z, this implies that X depends on Z (X $\rightarrow$ Z).

In Fig. 3. topological ordering is carried out and conflicts occur between nodes A2, A6 and A1, A5 which is shown in Table 1. To resolve these conflicts, second valid ordering is performed which in turn uses grey relational analysis, shows that A6 has a higher priority than A2, thus, A2 $\rightarrow$ A6, and A1 has a higher priority than A5, thus, A5 $\rightarrow$ A1 which is shown in Fig. 4. The topological order of the nodes is output when the graph is empty.

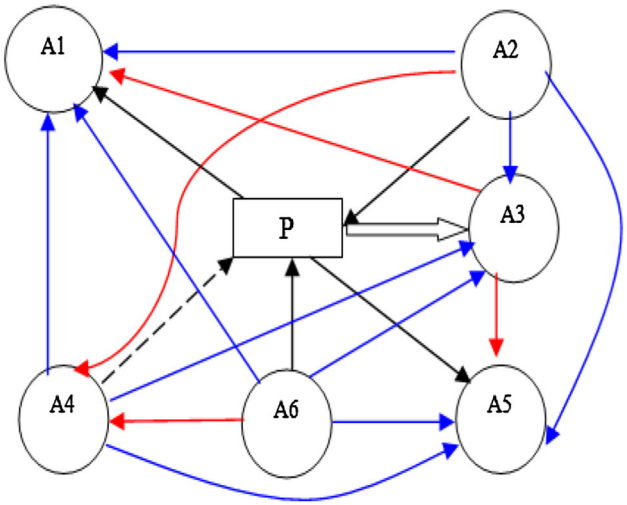


Fig. 3 Topological ordering of a DAG

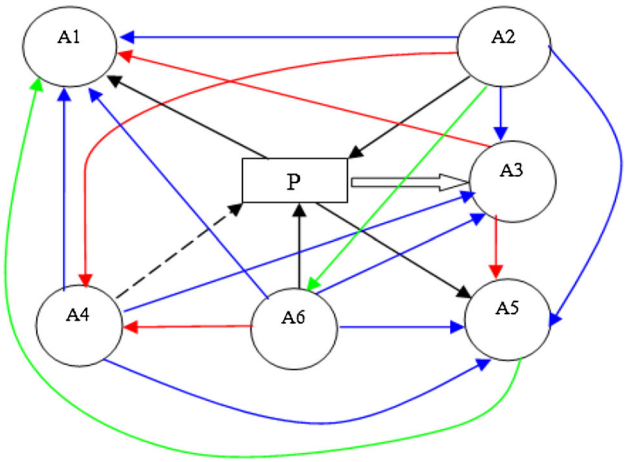


Fig. 4 Resolved DAG

The composition rules are created according to the direction of the small arrow as discussed in the topological ordering method given in [33].

Topological Ordering: A2A6A4PA3A5A1$\leftarrow$

Composition Rule: A1A5A3PA4A6A2

This means that A1 overlaps before P, A5 overlaps before P, A3 wraps P, A4 replaces P, A6 overlaps after P, and A2 overlaps after P.

5 Case Study

The Movie Theater Chain Portal needs a specially designed system to provide the vast functionality required for public access [15]. The system has been designed mainly to provide all the facilities for integrated online ticket sales. Among others, this system provides online facilities for buying tickets, reserving tickets, paying for tickets, and canceling tickets. Other important services provided by the system are checking

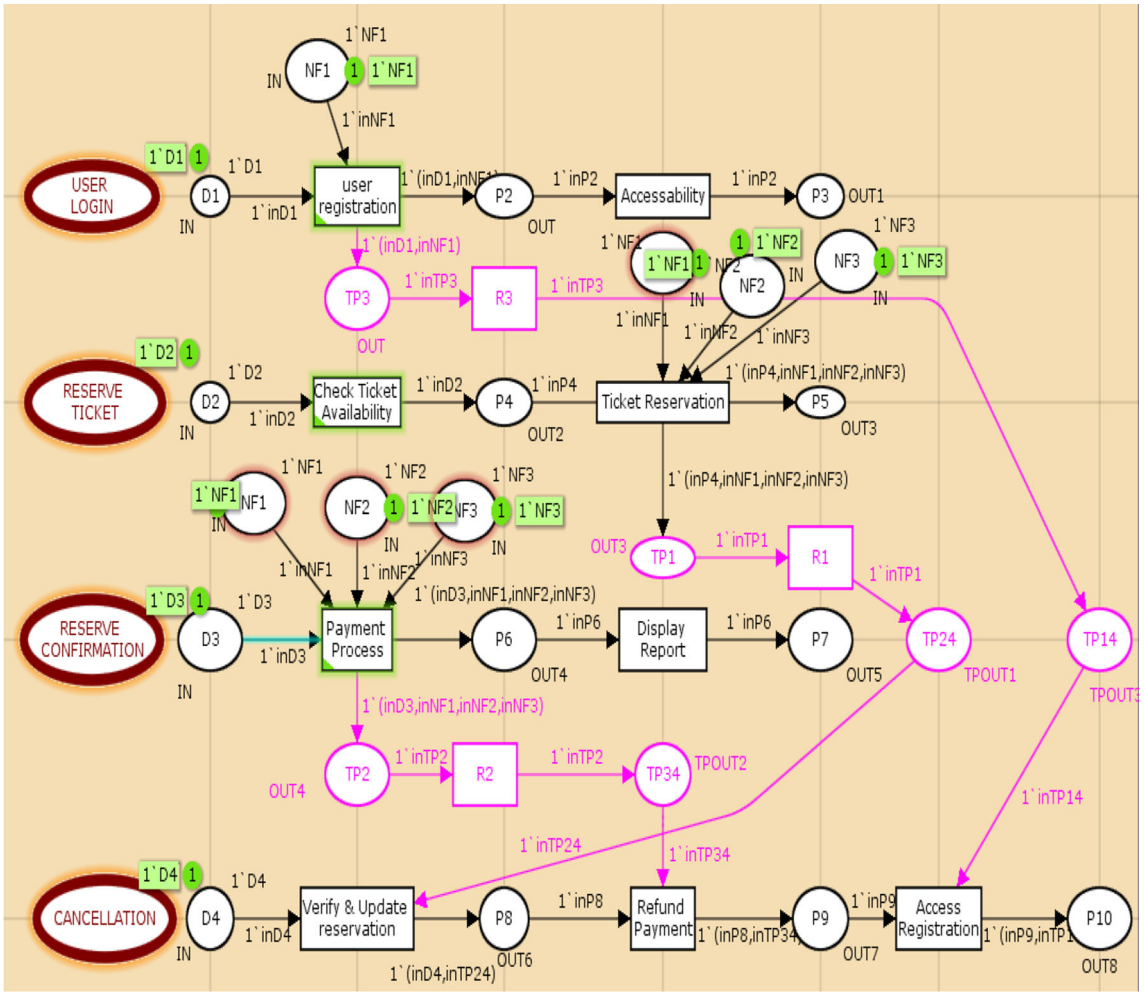


Fig. 5 Crosscutting analysis of functional and non-functional concerns

ticket availability, user registration, reserving, paying and canceling tickets by consultants, and transaction statistics. To identify the match points, we need to consider the functional and non-functional concerns presented in the system. In a system, match points usually occur where functional and non-functional concerns meet. In this system, the functional concerns are user registration, ticket booking, ticket payment, cancellation through consultation, and transaction statistics.

On the other hand, the non-functional concerns in this system are response time, precision, security, and reliability. Here, user registration involves security, ticket booking or reservation requires accuracy, response time, and security, and ticket payment requires security, response time, and accuracy.

By simulating the Movie Theater Chain Portal System using CPNs, the functional crosscutting concerns in this system are identified as payment and booking of tickets and user registration, while the non-functional concerns that crosscut the functional concerns are security, logging, accuracy, response time, availability, and precision which is shown in Fig. 5.

As the aspects are After, Before, Around, and Modify, the satisfaction of ticket payment depends on the satisfaction of ticket booking and registration. The dependencies identified are ticket payment → ticket booking and ticket payment → registration. Thus, ticket booking is processed as a before aspect with ticket payment, and similarly ticket registration (ticket booking) is processed as a before aspect with ticket booking and ticket payment in Fig. 5.

By taking into consideration operators such as Before, After, Around, and Replace, the composition specification of the match point ticket payment is given below:

- Response Time (RT) around Ticket Payment (TP): TP ⇒ RT
- Security. Availability (S.AV) before Ticket Payment: TP → S.AV
- Precision after Ticket Payment: Precision → TP

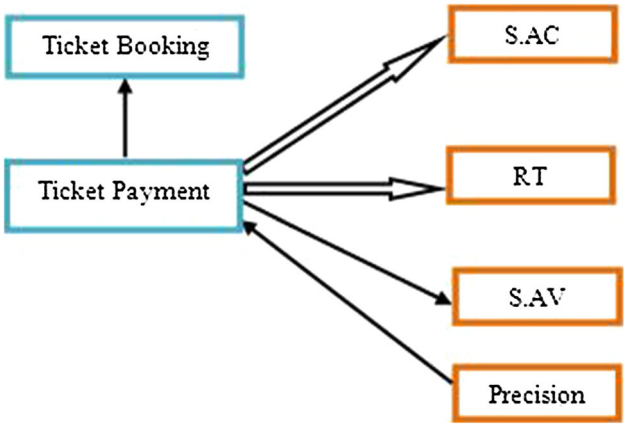


Fig. 6 Initial dependency graph

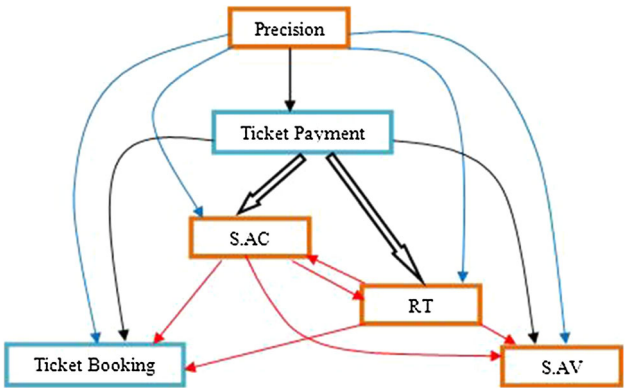


Fig. 7 Transitive dependency graph

- Security. Accuracy (S.AC) around Ticket Payment: TP ⇒ S.AC
- Ticket Booking (TB) before Ticket Payment: TP → TB

In the system, a fictive dependency is identified from the use of the Around operator. RT and Ticket Payment are parallel. So, RT is dependent on all aspects on which Ticket Payment depends.

RT → Ticket Booking; RT → S.AC; RT → S.AV.S.AC and Ticket Payment are parallel. So, S.AC is dependent on all aspects on which Ticket Payment depends. S.AC → Ticket Booking; S.AC → RT; and S.AC → S.AV.

Figure 6 shows the initial dependency graph for these fictive dependencies, from which transitive dependencies are identified as shown in Fig. 7.

A transitive relationship occurs between the aspects Precision, Ticket Booking, and Ticket Payment: Precision depends on Ticket Payment, and Ticket Payment depends on Ticket Booking, which implies that Precision depends on Ticket booking (Precision → Ticket Booking). The other transitive relationship that occurs is as follows: Precision depends on S.AC (Precision → S.AC), Precision depends on RT (Precision → RT), and Precision depends on S.AV (Precision → S.AV).

While processing these aspects, we found that the graph was not a DAG. Therefore, the feedback edge set method was used to eliminate the edge between S.AC and RT (S.AC → RT) since it is a weaker dependency. Thereafter, topological ordering was applied to the resolved DAG, yielding an ordering of the aspects which is shown in Table 2.

In the obtained ordering, if there are any conflicts between the aspects, a second valid ordering approach is used. We compare each element in the first column of the adjacency matrix with the remaining column elements. If no column is the same as the first column, there are no conflicts between the first and remaining aspects. However, it is possible that a column is the same as the previous column, in which case, a conflict arises between them. By repeating the procedure with the remaining columns, we can identify all such conflicts. In the case that conflicts exist, grey relational analysis is carried out [30–35]. In our case study on the Movie Theater Chain Portal System, conflicts exist between S.AV and TB, and thus, grey relational analysis is applied as follows. The data processing steps for grey relational analysis are given below.

Step 1: Create a standard array and comparison arrays. We give the details for calculating grey relations using the results of integrating the five stakeholder groups in the example [36].

a. Standard array $X_0(k) = \{10, 10, 10, 10, 10, 10, 10\}$, the full credit "10" is set as the standard array which is shown in Table 3.

b. Questionnaire on the concerns for the movie theater chain portal system is shown in Table 4.

Comparison arrays can be calculated by integrating the five stakeholders groups which is shown in Table 5.

Comparison arrays:

$$X1 = \{6, 4, 5, 5, 3\}$$
$$X2 = \{8, 7, 9, 7, 9\}$$
$$X3 = \{9, 9, 8, 9, 9\}$$
$$X4 = \{7, 8, 9, 6, 7\}$$
$$X5 = \{6, 7, 8, 7, 6\}$$
$$X6 = \{7, 6, 7, 5, 6\}$$

The standard and comparison arrays are represented as $X_0(k)$ and $X_i(k), i = 1, 2, \ldots, m; k = 1, 2, \ldots, n$, respectively, where m is the total number of aspects and n is the total number of stakeholders.

Step 2: The difference arrays Δ_{0i} are calculated as

$$\Delta_{0i} = |X_O(k) - X_i(k)|$$
$$\Delta_{01} = \{4, 6, 5, 5, 7\}$$
$$\Delta_{02} = \{2, 3, 1, 3, 1\}$$
$$\Delta_{03} = \{1, 1, 2, 1, 1\}$$
$$\Delta_{04} = \{3, 2, 1, 4, 3\}$$
$$\Delta_{05} = \{4, 3, 2, 3, 4\}$$

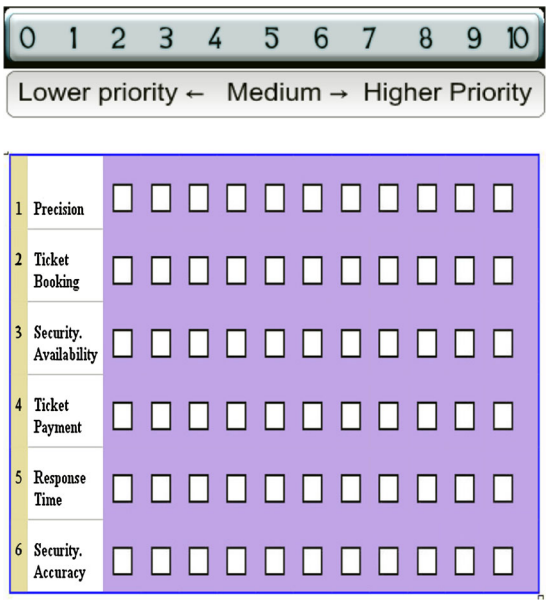
Table 2 Topological ordering process

Vertices	In-degree						
S.AV	4	3	2	1	0	–	–
TP	1	0	–	–	–	–	–
P	0	–	–	–	–	–	–
S.AC	3	2	1	0	–	–	–
RT	2	1	0	–	–	–	–
TB	4	3	2	1	0	–	–
Enqueue	P	TP	RT	S.AC	S.AV,TB	TB	Nil
Dequeue	P	TP	RT	S.AC	S.AV	TB	Nil

Table 3 Relationship between functional and non-functional aspects specified by stakeholders in the Movie Theater Chain Portal System

Non-functional and functional aspects	Standard
Precision	10
Ticket Booking	10
Security. Availability	10
Ticket Payment	10
Response Time	10
Security Accuracy	10

Table 4 Questionnaire on the concerns for the Movie Theater Chain Portal System



$\Delta_{06} = \{3, 4, 3, 5, 4\}$.

From $\Delta_{\max} = \max_{\forall j \in i} \max_{\forall k} |X_0(k) - X_i(k)|$, the maximum value in the difference arrays is Max = 7, and from $\Delta_{\min} = \min_{\forall j \in i} \min_{\forall k} |X_0(k) - X_i(k)|$ the minimum value in the difference arrays is Min = 1. The distinguishing coefficient value ς is 0.5, where $\varsigma \in [0, 1]$.

Step 3: The grey relational coefficient $\gamma_{0i} = \gamma(X_0(k), X_i(k))$ values are calculated as

$$\gamma(X_o(k), X_i(k)) = \frac{\Delta_{\min} + \varsigma \Delta_{\max}}{\Delta_{0i}(k) + \varsigma \Delta_{\max}}$$

$$0 \prec \gamma(X_0(k), X_i(k)) \leq 1.$$

$$X_1 : \gamma(X_0(1), X_1(1)) = 0.6\gamma(X_0(2), X_1(2)) = 0.47$$
$$\gamma(X_0(3), X_1(3)) = 0.53\gamma(X_0(4), X_1(4)) = 0.53$$
$$\gamma(X_0(5), X_1(5)) = 0.43$$
$$X_2 : \gamma(X_0(1), X_2(1)) = 0.81\gamma(X_0(2), X_2(2)) = 0.69$$
$$\gamma(X_0(3), X_2(3)) = 1\gamma(X_0(4), X_2(4)) = 0.69$$
$$\gamma(X_0(5), X_2(5)) = 1$$
$$X_3 : \gamma(X_0(1), X_3(1)) = 1\gamma(X_0(2), X_3(2)) = 1$$
$$\gamma(X_0(3), X_3(3)) = 0.81\gamma(X_0(4), X_3(4)) = 1$$
$$\gamma(X_0(5), X_3(5)) = 1$$
$$X_4 : \gamma(X_0(1), X_4(1)) = 0.69\gamma(X_0(2), X_4(2)) = 0.81$$
$$\gamma(X_0(3), X_4(3)) = 1\gamma(X_0(4), X_4(4)) = 0.6$$
$$\gamma(X_0(5), X_4(5)) = 0.69$$
$$X_5 : \gamma(X_0(1), X_5(1)) = 0.6\gamma(X_0(2), X_5(2)) = 0.69$$
$$\gamma(X_0(3), X_5(3)) = 0.81\gamma(X_0(4), X_5(4)) = 0.69$$
$$\gamma(X_0(5), X_5(5)) = 0.6$$
$$X_6 : \gamma(X_0(1), X_6(1)) = 0.69\gamma(X_0(2), X_6(2)) = 0.6$$
$$\gamma(X_0(3), X_6(3)) = 0.69\gamma(X_0(4), X_6(4)) = 0.53$$
$$\gamma(X_0(5), X_6(5)) = 0.6$$

Step 4: Calculate the value of the grey relation

$$\gamma(X_0, X_i) = \frac{1}{n} \sum_{k=1}^{n} \gamma(X_0(k), X_i(k)).$$

$$\gamma(X_0, X_1) = (1/5)(2.56) = 0.512$$
$$\gamma(X_0, X_2) = (1/5)(4.19) = 0.838$$
$$\gamma(X_0, X_3) = (1/5)(4.81) = 0.962$$
$$\gamma(X_0, X_4) = (1/5)(3.79) = 0.758$$
$$\gamma(X_0, X_5) = (1/5)(3.39) = 0.678$$
$$\gamma(X_0, X_6) = (1/5)(3.11) = 0.622$$

Table 5 Outcome of integrating the five stakeholder (SH) groups

Non-functional and functional aspects	SH group 1	SH group 2	SH group 3	SH group 4	SH group 5
Precision	6	4	5	5	3
Ticket Booking	8	7	9	7	9
Security. Availability	9	9	8	9	9
Ticket Payment	7	8	9	6	7
Response Time	6	7	8	7	6
Security Accuracy	7	6	7	5	6

Table 6 Final ranking of aspects in Movie Theater Chain Portal System

Non-functional and functional aspects	Grey relational values	Ranking order
Security. Accuracy	0.622	5
Ticket Payment	0.758	3
Security. Availability	0.962	1
Ticket Booking	0.838	2
Response Time	0.678	4
Precision	0.512	6

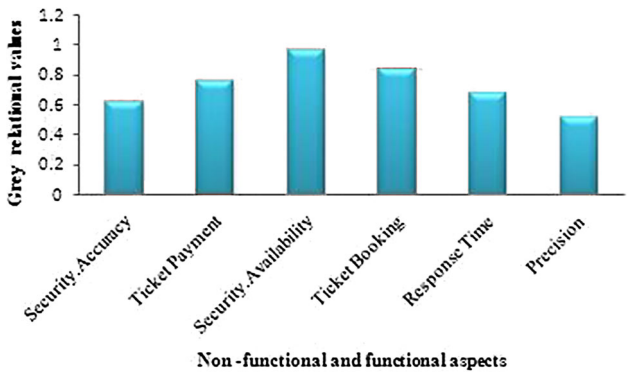


Fig. 8 Comparison of grey relational values between aspects

Step 5: Arrange the grey relational values in decreasing order:

$0.962 > 0.838 > 0.758 > 0.678 > 0.622 > 0.512$

Thus, the final ranking of aspects is:
S.AV > Ticket Booking > Ticket Payment > S.AC > RT > Precision. This shows that S.AV has a higher priority than TB since TB depends on S.AV (TB → S.AV). According to this ordering, the conflicting aspects are arranged in order by swapping them (Table 6; Fig. 8).

Topological order: P TP RT S.AC S.AV TB ←
Grey relational order: S.AV > TB > TP > S.AC > RT > P
Composition Rule: S.AV TB S.AC RT TP P

Table 7 ANOVA for aspects and stakeholders

S.V	SS	D.F	M.S	F
Between aspects	50.7	5	10.14	10.67
Between stakeholders	5	4	1.25	1.32
Residual	19	20	0.95	–

Using the FTS approach, the composition rule obtained for the match point Ticket Payment in the Movie Theater Chain Portal is represented using LOTOS operators [33] as: S.AV ≫ Ticket Booking ≫ (Ticket Payment || S.AC || RT) ≫ Precision. Additionally, Table 7 gives the results of the analysis of variance (ANOVA) for the aspects and stakeholders using the experiential values [37]. According to this table, there is no significant difference among stakeholders, but there is a significant difference between aspects.

This composition rule expresses the sequential order in which each candidate aspect must be composed. Availability, a sub-concern of security must be satisfied first before the ticket booking aspect. Then, ticket payment needs to be fulfilled in parallel with accuracy, a sub-concern of security, and response time. Finally, precision must be satisfied. The time complexity of the second valid ordering algorithm is calculated as

$$T(n,n) = \sum_{i=1}^{n-1} n*(n-i)$$

$$T(n,n) = \mathrm{O}\left(n^3\right)$$

where n represents the number of nodes in the graph. Hence, the time complexity for the second valid ordering algorithm is $\mathrm{O}(n^3)$ and the time complexity for topological ordering is $\mathrm{O}(V+E)$. Therefore, considering the properties of Big O notation, the overall time complexity of the FTS approach is $\mathrm{O}\left(n^3\right)$. The time complexity of the longest path algorithm, which is used by the existing method, is given by O(n*n!). The recurrence relation for the longest path algorithm is given by $T(n \cdot m) = mT(n, m-1) + O(n)$, where n denotes the number of nodes in the graph, m denotes the number of

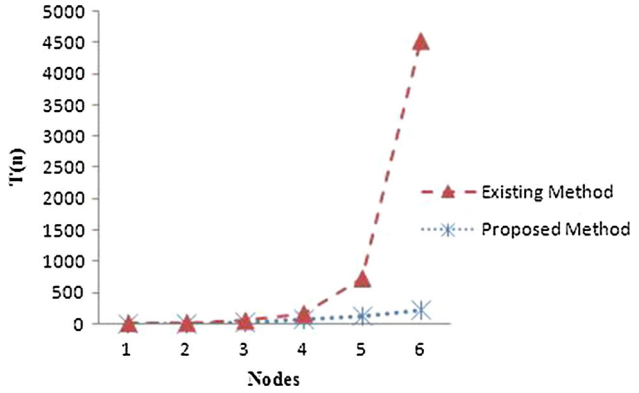


Fig. 9 Performance analysis

unvisited nodes in the graph and $T\ (n, 0) = \mathrm{O}\ (n)$.

$$T\ (n, n) = nT\ (n, n-1) + \mathrm{O}\ (n)$$
$$T\ (n, n) = \mathrm{O}\ (n{*}n!)$$

From the performance analysis graph of time complexity in Fig. 9, we can see that the execution performance of the proposed approach is superior to existing approaches with the number of vertices greater than or equal to four.

6 Conclusion

Our proposed framework describes the role of CPNs in identifying aspects during requirements engineering itself and imposing constraints on the aspects (*After*, *Before*, *Around,* and *Replace*) by providing new methods for the identification, specification, and composition of crosscutting concerns used in our FTS approach. This approach minimizes the time complexity when considering four or more vertices by creating a composition rule from the composition specifications, which requires $O(n^3)$ time rather than O(n*n!). Hence, our proposed approach FTS provides a better performance. In future work, we intend to focus on the following tasks: defining composition rules at a finer level of granularity, i.e., composing crosscutting actions; and studying the level of granularity at which conflicting situations can be handled.

References

1. Cancila, D.; Passerone, R.; Vardanega, T.; Panunzio, M.: Toward correctness in the specification and handling of non-functional attributes of high-integrity real-time embedded systems. IEEE Trans. Ind. Inform. **6**(2), 181–194 (2010). doi:10.1109/TII.2010.2043741
2. Abdelzad, V.; Aliee, F.S.: A method based on petri nets for identification of aspects. Inf. Sci. Technol. Bull. ACM Slovak. **2**(1), 43–49 (2010)
3. Ali, R.; Dalpiaz, F.; Giorgini, P.: Reasoning with contextual requirements: detecting inconsistency and conflicts. Inf. Softw. Technol. **55**(1), 35–57 (2013). doi:10.1016/j.infsof.2012.06.013
4. Boubendir, A.; Chaoui, A.: On analyzing interactions between aspects at requirements phase. J. Theor. Appl. Inf. Technol. **18**(2): JATIT & LLS (2010a), http://www.jatit.org/volumes/research-papers/Vol18No2/3Vol18No2.pdf
5. Boubendir, A.; Chaoui, A.: Towards a generic technique for analyzing interactions between aspects at requirement phase. In: Digital Information Management (ICDIM), Canada, pp. 507–512 (2010b). doi:10.1109/ICDIM.2010.5664647
6. Mehner, K.; Monga, M.; Taentzer, G..: Interaction analysis in aspect-oriented models. In: 14th IEEE International Requirements Engineering Conference (RE'06), pp. 69–78 (2006). doi:10.1109/RE.2006.35
7. Zayaraz, G.; Thambidurai, P.D.; Srinivasan, M.; Rodrigues, P.D.: Software quality assurance through COSMIC FFP. ACM SIGSOFT Softw. Eng. Notes **30**(5), 1–5 (2005). doi:10.1145/1095430.1095445
8. Shen, H.; Dorina, C.; Petriu.: Performance analysis of UML models using aspect-oriented modeling techniques. In: 8th International Conference MoDELS 2005, pp 156–170 (2005).
9. Beniassad, E.; Clements, P.C.; Araujo, J.; Moriera, A.; Rachid, A.; Tekmerdogan, B.: Discovering early aspects. IEEE Softw. **1**, 61–70 (2006). doi:10.1007/978-1-4614-33637
10. Brito, I.; Moreira, A.: Towards a composition process for aspect-oriented requirements. In: Proceeding of AOSD'03 Workshop on Early Aspects: Aspect Oriented Requirements Engineering and Architecture, March 17, Boston USA (2003)
11. Rachid, A., Moreira, A., Araujo, J.: Modularization and composition of aspectual requirements. In: 2nd International Conference on Aspect Oriented Software Development (AOSD), pp. 11–20, ACM, Boston, USA (2003). doi:10.1145/643603.643605
12. Alshayeb, M.: The impact of refactoring to patterns on software quality attributes. Arab. J. Sci. Eng. **36**(7), 1241–1251 (2011). doi:10.1007/s13369-011-0111-3
13. Zhang, L.; Feng, S.: Aspect-oriented QoS modeling of cyber-physical systems by the extension of architecture analysis and design language. In: Computer Engineering and Networking, pp. 1125–1131. Springer International Publishing (2014). doi:10.1007/978-3-319-01766-2_128
14. Fuentes, L.; Sanchez, P.: Towards executable aspect-oriented UML models. In: Proceedings of the 10th International Workshop on Aspect-Oriented Modeling, New York, pp. 28–34 (2007). doi:10.1145/1229375.1229380
15. Susanne, C.: UML extensions for aspect oriented software development. J. Object Technol. **8**(5), 85–104 (2009)
16. Douance, R.; Frader, P.: Detection and Resolution of Aspect Interactions. INRIA Technical Report N°RR 4435 (2002)
17. Binder, W.; Ansaloni, D.; Villazón, A.; Moret, P.: Flexible and efficient profiling with aspect-oriented programming. Concurr. Comput. Pract. Exp. **23**(15), 1749–1773 (2011). doi:10.1002/cpe.1760
18. Cemus, K.; Cerny, T.: Aspect-driven design of information systems. In: SOFSEM 2014: Theory and Practice of Computer Science, pp. 174–186. Springer International Publishing (2014). doi:10.1007/978-3-319-04298-5_16
19. Zhang, L.: QoS modeling of cyber physical systems by the integration of AADL and aspect-oriented methods. In: Advanced Technologies, Embedded and Multimedia for Human-Centric Computing, pp. 419–428. Springer Netherlands (2014). doi:10.1007/978-94-007-7262-5_49
20. Santhi, K.; Zayaraz, G.; Vijayalakshmi, V.; Santhi, K.; Zayaraz, G.; Vijayalakshmi, V.: Aspect-oriented analyzer framework for aspect oriented requirements. In: Proceedings of the 3rd International Conference on Trends in Information, Telecommunicating

and Computing Lecture Notes in Electrical Engineering, vol. 150, pp. 75–81 (2013). doi:10.1007/978-1-4614-3363-7_9
21. Santhi, K.; Zayaraz, G.: An approach based on colored petri net for analysing and modelling the aspects. Int. J. Hybrid Inf. Technol. **6**(5), 25–36 (2013). doi:10.14257/ijhit.2013.6.5.03
22. Lianwei, G.; Li, X.; Hu, H.: Petri net-based approach for supporting aspect oriented modeling. In: 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering (2008). doi:10.1109/TASE.2008.32
23. Kahn, A.B.: Topological sorting of large networks. Commun. ACM **5**, 558–562 (1962). doi:10.1145/368996.369025
24. Silveira, F.F.; da Cunha, A.M.; Lisbôa, M.L.: A state-based testing method for detecting aspect composition faults. In: Computational Science and Its Applications—ICCSA 2014, Lecture Notes in Computer Science, vol. 8583, pp. 418–433 (2014)
25. Broy, M.: Multifunctional software systems: structured modeling and specification of functional requirements. Sci. Comput. Program. **75**(12), 1193–1214 (2010). doi:10.1016/j.scico.2010.06.007
26. Sofian, H.B.; Salim, S.S.B.; Shahamiri, S.R.: A requirements negotiation process model that integrates EasyWinWin with quality assurance and multi-criteria preference technique. Arab. J. Sci. Eng. **39**(6), 4667–4681 (2014). doi:10.1007/s13369-014-1150-3
27. Elish, K.; Alshayeb, M.: A classification of refactoring methods based on software quality attributes. Arab. J. Sci. Eng. **36**(7), 1253–1267 (2011). doi:10.1007/s13369-011-0117-x
28. Saberi, S.; Nookabadi, A.S.; Hejazi, S.R.: Applying agent-based system and negotiation mechanism in improvement of inventory management and customer order fulfilment in multi echelon supply chain. Arab. J. Sci. Eng. **37**(3), 851–861 (2012). doi:10.1007/s13369-012-0197-2
29. Lee, S.-H.; Yoo, H.: Requirement analysis for aspect-oriented system development. In: IT Convergence and Security 2012, Lecture Notes in Electrical Engineering, vol. 215, pp. 1201–1209 (2013). doi:10.1007/978-94-007-5860-5_144
30. Chan, W.K.; Tong, T.K.L.: Multi-criteria material selections and end-of-life product strategy: grey relational analysis approach. Mater. Des. **28**(5), 1539–1546 (2007). doi:10.1016/j.matdes.2006.02.016
31. Feng, L.,; Yuan, R.: Study on grey relation analysis based on entropy method in evaluation of logistics center location. In: Third International Conference on Measuring Technology and Mechatronics Automation, vol. 3, pp. 474–477. doi:10.1109/ICMTMA.2011.689
32. Rao, R.V.; Singh, D.: An improved grey relational analysis as a decision-making method for manufacturing situations. Int. J. Decis. Sci. Risk Manag. **2**(1/2), 1–23 (2010). doi:10.1504/IJDSRM.2010.034668
33. Logrippo, L.; Faci, M.; Haj-Hussein, M.: An introduction to LOTOS: learning by examples. Comput. Netw. ISDN Syst. 23(5), 325–342 (1992). doi:10.1016/0169-7552(92)90011-E
34. Mehat, N.M.; Kamaruddin, S.; Othman, A.R.: Hybrid integration of taguchi parametric design, grey relational analysis, and principal component analysis optimization for plastic gear production. Chin. J. Eng. 2014, Article ID 351206, 11 p (2014). doi:10.1155/2014/351206
35. Xie, Y.; Chengdu. Yin, S.; Luo, Z.: Robust optimization for deep-drawing process of sheet metal based on CAE with grey relational analysis method. In: Fifth International Conference on Fuzzy Systems and Knowledge Discovery, vol. 4, pp. 345–349 (2008). doi:10.1109/FSKD.2008.418
36. Yu, Y.-L.; Cho, L.-C.; Liao, B.-T.: The influenced factors in buying cell phones by grey relational analysis. WSEAS Trans. Inf. Sci. Appl. **4** (6), 1151–1156 (2013), http://140.128.103.12/handle/310901/22021
37. Goel, P.; Khan, Z.A; Siddiquee, A.N.; Kamaruddin, S.; Gupta, R.K.: Influence of slab milling process parameters on surface integrity of HSLA: a multi-performance characteristics optimization. Int. J. Adv. Manuf. Technol. **61**(9-12), 859–871 (2012). doi:10.1007/s00170-011-3763-y