Software Requirements Conflict Identification: Review and Recommendations

Maysoon Aldekhail
College of Computer and Information Sciences
King Saud University

Azzedine Chikh
College of Computer and Information Sciences
King Saud University

Djamal Ziani
College of Computer and Information Sciences
King Saud University

***Abstract*—Successful development of software systems requires a set of complete, consistent and clear requirements. A wide range of different stakeholders with various needs and backgrounds participate in the requirements engineering process. Accordingly, it is difficult to completely satisfy the requirements of each and every stakeholder. It is the requirements engineer's job to trade-off stakeholders' needs with the project resources and constraints. Many studies assert that failure in understanding and managing requirements in general, and requirement conflicts in particular, are one of the main problems of exceeding cost and allocated time which in turn results in project failure.**
This paper aims at investigating the different reasons of requirements conflicts and the different types of requirements conflicts. It providing an overview of existing research works on identifying conflicts; and discussing their limitations in order to yield suggestions for improvement.
Objective: To provide an overview of existing research studies on identifying software requirements conflict and identifying limitations and areas for improvement.
Method: A comparative literature was conducted by assessing 20 studies dated from 2001 to 2014.

Keywords*—software requirements; requirements engineering; requirements conflicts

I. Introduction

In requirement engineering, the term conflict involves interference, interdependency or inconsistency between requirements [1].

Different studies state that failure in managing requirement conflicts is one of the main reasons for failure in software projects which is caused by cost and lack of time [2]. It is essential to detect and resolve conflicts in early phases of the project lifecycle to prevent re-iterations of all phases [3]. In recent research studies, a high number of conflicting requirements is stated as in [4], n^2 conflicts are reported in n requirements, whereas [5] reported 40%-60% of requirements were in conflict, In addition, the functional and nonfunctional requirements were both found to be equal in the percentage of conflicts.

Also, most research has shown the risks of working with requirements that are in conflicts with other requirements. These risks are overtime or over budget which can lead to project failure. At the very least, it would result in extra effort expended.

The remainder of the paper is organized as follows: section II gives an overview about requirements conflict, the different reasons for requirements conflict and the different types of requirements conflict. Section III presents in details the existing techniques for requirements conflict and a comparison between them. Then, section IV, discusses the limitation and research gaps in previous works and gives some recommendations should be taken into consideration when working to find practical techniques for detecting conflict between requirements. Finally, a conclusion of the review is giving.

II. Requirements Conflict

This section explains the meaning of requirements conflicts, the different reasons that may cause conflict between requirements and the different types of requirements conflicts.

A. Definition of Requirements Conflict

Conflicting requirements is a problem that occurs when a requirement is inconsistent with another requirement [7]. Consistency between requirements requires no two or more requirements contradict each other [8]. In requirements engineering, the term conflict involves interference, interdependency or inconsistency between requirements [1].

Kim et al. [9] gave a good definition of requirements conflict as:

"The interactions and dependencies between requirements that can lead to negative or undesired operation of the system" An example of a conflict in nonfunctional requirements can be the gap between performance and security; when the client wants certain functionality to be satisfied in minimal time (e.g. calculate something and display it on screen), as well as the use of a secure protocol for data transferee and double password access control.

B. Causes of Requirements Conflict

There are different reasons that cause conflicts between stakeholders' requirements. One good categorization for conflicts reasons is presented in [10]; it classifies the reasons into technical reasons and social reasons. Technical reasons are caused by the following difficulties:

- Massive quantity of requirements can lead to conflicts between them.
- Changes in requirements during system development phases. These changing may occur after the addition of new requirements or the update of old ones [14].

- Complex system domain can lead to misunderstanding of requirements, and therefore, conflicts between them.

Whereas, the social difficulties that lead to requirements conflicts are as follows:

- System has different stakeholders with diverse interests that usually interact with each other and causes conflicts.
- Changes in the system's stakeholders by adding new stakeholders with different needs or by changing stakeholders' requests.

Therefore, there are different sources for inconsistencies between requirements and these may cause problems in the success of the software development. Researchers have been working to find various solutions for this problem.

C. Types of Requirements Conflict

The literature review has shown that there are no predefined classifications for conflicts in requirements. Each work provides a different classification for the conflicts after its found based on the technique used to detect conflicts.

Poort and de With [15] grouped functional requirements based on nonfunctional requirements; this means finding all primary function requirements that share similar nonfunctional requirements and grouping them together. Then two types of conflicts are defined: grouping conflicts which caused by differences in grouping of functions and in-group conflicts that have conflicting requirements within one function group. For example, there are three function groups called workflow, data entry and analysis. For data entry and analysis, security requirements are more restrictive than in the workflow group. Whereas, modifiability for analysis are more stringent than those for data entry and work flow.

Sadana and Liu [16] analyzed functional and nonfunctional requirements and built functionality and quality attributed to hierarchy. Then, two types of conflict in NFR are defined based on comparison of all the lowest level NFRs, if there is still a conflict detected among the NFR. These types are mutually exclusive and partial conflicts. Mutually exclusive conflicts as follows: NFRs A, B are mutually exclusive conflicts if all the lowest level requirements in NFR A have a conflict with the lowest level requirements in NFR B. Partial conflicts as follows: NFRs A, B are in partial conflicts if some of the lowest level requirements in NFR A have a conflict with the lowest level requirements in NFR B.

Heng and Ming [17] defined three types of inconsistent requirements based on multi-coordinated views on requirements. This is common when one stakeholder has an incomplete requirement while others have complete requirements. This creates a situation where requirements overlap when one part of processes in set A is overlapped with another but not fully overlapped, as well as totally disjointed requirements when two views on requirements are totally disjointed.

Butt et al. [2] defined different conflicts based on the classification of the requirements to mandatory, essential and optional. Mandatory requirements are a set of functional and nonfunctional requirements. Essential requirements are the constraints of the mandatory requirements. Whereas optional requirements are the requirements that if they have conflicts, this would not affect the acceptance of the system. For example, in a hostel management system for a university:

- The system should allow the warden to assign student a seat in his hostel (Mandatory requirement).
- The system should maintain a log of all allotments and vacations in his hostel (Essential requirement).
- The system should allow the warden to shuffle multiple students seats (Optional requirement).

Kim et al. [9] defined two types of requirement conflicts depending on the cause of the conflict and the authoring structure, which is action (verb) + object (object) + resource (resource):

- Source conflict when two requirements use the same resource.

For example, with cellular phones, when a phone call is made from a number that should not answered , the *automatic response function* will try to answer the phone call while the *reception refusal function* will be forced not to answer the call.

- Activity conflict either by opposite verb (verb (different) +object (same)) or by different object (verb (same) +object (different)).

For example, a fire control function is required with an intrusion control function in the home integration system. When those two functions are executed simultaneously, they will try to send messages (fire message and intrusion message) using the same resource (telephone service) at the same time, which will lead to a resource conflict.

Moser et al. [13] [18] defined three types of conflicts that could be detected: conflict between a requirement and a constraint (CRC), conflict between a requirement and a guideline (CRG), and conflict between requirements (CRR). They also gave two classifications for conflicts based on the number of requirements, simple conflicts (between two requirements), and complex conflicts (between three or more).

Urbieta et al. [19] [20] defined three types of conflicts on Web application:

- Structural conflicts: Which means the difference in the data is expected to be presented on a Web page by different stakeholders
- Navigational conflicts: This occurs when two Web application requirements may contradict the way in which links are traversed which in turn produces navigational conflicts; that is, having two targets go to a single source.
- Semantic conflicts: this happens when the same real-world object is described using different terms.

Chentouf [21] defined seven types of conflicts:

1) Duplicated requirements: If two requirements are exactly the same or one is included in the other.

2) Incompatible requirements: If two requirements are either ambiguous, incompatible or contradictory.
   a) Two operation frequencies: when the same agent is required to perform the same operation on the same object, but at two different frequencies.
   b) Start-forbid: when the same event causes the same operation to be performed and forbidden.
   c) Forbid-stop: when the same operation is stopped under a certain condition event and at the same time, is unconditionally forbidden in another requirement.
   d) Two condition events: when the same operation is being executed, stopped or forbidden on two different events.
3) Assumption alteration: when the output of one requirements' operation is part of the inputs (assumptions) or outputs (results) of the other's operation.
   a) Input-output: When one of the requirements performs its operation on an object (output) that is an input in other requirements.
   b) Out-put: This happens if one requirement alters the result (output) or part of another requirement.

Mairiza and Zowghi [21] explained the different categories of conflicts in NFRs as:

- Absolute conflict: represents a pair of NFRs types that are always in conflict.
  For example, security and performance, availability and privacy.
- Relative conflict: represents a pair of NFRs that are claimed to be in conflict in some cases but not in all. For example, usability and security, usability and performance.
- Never conflict: represents a pair of NFRs types that in the software projects are never inconflict. They may contribute either positively through support or cooperation, or may be indifferent to one another.
  For example, accuracy and security, usability and maintainability.

In general, we can give general classifications to requirements conflicts based on the types of requirements, functional requirements and nonfunctional requirements. An example for conflicts in nonfunctional requirements is security (privacy metric) with usability (ease of function learning metric) so there is a tradeoff between them. Then the developer must choose a satisfactory solution to find the right balance of attributes that work. Another example is:

- R1: After three continues failed login attempts, the account would be locked by the system.
- R2: Once the account is locked, the system sends an account lock notification email to the account's owner.
- R3: Once an account is locked, the system would also send a SMS message to the account's owner to notify him about the situation owner.
- R4: If a user has already received a notification via email, he will not receive the same notification via SMS.
- There is a conflict between R2, R3 and R4.

Another good classification for requirements conflicts is the one illustrated in [21].

III. Requirement Conflict Identification Techniques

Owing to the importance of accurate and complete requirements, researchers have tried to identify detection techniques and proposed solutions for requirements conflicts.

This section discusses the different existing detection techniques and their categorization. In the end, a comparison and analysis of the techniques is summarized in a table.

The techniques proposed can have different classifications; the easiest classification is the negotiation or automation techniques. In negotiation techniques, stakeholders and software engineers manually discuss and analyze requirements to detect any conflicts [13]. Some call this approach an informal technique that can be achieved by hiring experts to detect inconsistencies using their experience [22]. This method has some disadvantages because it may take a long time and much effort to negotiate between different stakeholders. Additionally, hiring experts can be very expensive and leave the process to be prone to errors. While in automation approaches, software engineers can use some tools to help with analyzing and managing requirements [13].

In [6], three approaches are proposed to detect requirements conflicts. The ontological approach which uses ontology to extract conflicts between terms and then, between requirements. The methodological approach compares requirement representations to find conflicts and resolve them. The technological approach provides a specific technique or automation to detect potential conflicts.

Methodological approach is almost the same as the negotiation approach since both are manual processes and depend on human efforts. Additionally, technological approach is similar to the automation technique since they both utilize tools to solve problem of requirements conflicts.

Another classification of current detection approaches are formalization-based approaches, model-based approaches and stakeholder priority approaches [23]. The formalization-based approaches use formal specifications for requirements to support seeking conflicts between them. The drawbacks of this approach are the time and effort needed to formalize the requirements and any mistake that could occur during the formalization may lead to incorrect conflict detection. The model-based approach structures the requirements into specific models before conflict identification. If the approach uses a model that is already used in the system then developing it is fine; however, if it uses a different model, this will create additional steps and therefore, extra time and effort. The third approach depends on the stakeholders' discussion and the stakeholders preferences.

A. Existing Techniques

Literature shows that requirements engineering is one of the most active research fields in the recent years. Researchers are continuously working to improve requirements quality and to resolve difficulties that may affect requirements wholeness or accuracy. One of the most common problems is requirements conflicts, and because of the importance of this topic as mentioned in section B, many works have presented different techniques to detect and resolve conflicts between requirements.

This section discusses these techniques, which can be placed in three categories:

1) Manual techniques done manually by requirement engineers.
2) Automatic techniques applied automatically using software tools.
3) General framework, to detect conflicts without using special techniques.

The different techniques are presented in ascending order based on their dates.

1) Manual: Most of the proposed methods are performed manually with software engineers and with help of stakeholders. Heisel and Souquierers [3] presented a heuristic algorithm to detect feature interactions in requirements. The algorithm uses the schematic versions for formalized requirements and consists of two parts, precondition interaction analysis to determine any two requirements where both might be applied. Then postcondition interaction analysis to determine the candidate incompatible requirements. As the algorithm is named 'heuristic', the candidates need to check with the software engineers and stakeholders to determine if they are actual conflicts or not.

Robinson [6] used a root requirements analysis to detect requirements interactions. The technique is composed of three procedures. First, rewrite the requirements in structure form. Then, produce the root requirements hierarchies. Finally, analyze the root requirements to determine the ordering of the requirements according to their degree of expected conflict. The case study result demonstrated that using root requirement analysis is more accurate and detects more conflicts than without using root analysis.

Poort and de With [15] presented a non-functional decomposition (NFD) model that gives a new classification for requirements. Primary functional requirements and supplementary requirements which is classified as secondary functional requirements, quality attribute requirements and implementation requirements.

The technique defined two types of conflict: grouping conflict caused by differences in grouping of functions and in-group conflicts when conflict happen within one function group. To solve in-group conflicts, requirements will be split into different functions. The new functions will be included in other function groups. This process will repeat until there is no in-group conflict found.

Sadana and Liu [16] have proposed a framework to analyze the conflicts among nonfunctional requirements using the integrated analysis of functional and non-functional requirements. The conflict detection is performed on the high level NFR based on the relationship between quality attributes, constraints and functionality. The FR and NFR hierarchy are built and integrated to produce high level NFR.

The conflict detection in NFRs is based on relationship among ISO 9126 quality attributes. Two types of conflict in NFR are defined mutually exclusive and partial conflict.

Liu [13] utilized an ontological approach to analyze conflicts in the requirements specification of activity diagrams. The requirements conflict process starts by building an action state ontology and drawing the activity diagram for existing requirements. Then, it detects the requirements conflict based on seven proposed rules: shortcut conflict, initial state conflict, final state conflict, sequence conflict, action state addition conflict, action state deletion conflict and process length conflict.

Heng and Ming [17] proposed a non-mathematical technique called multi-coordinated views that showing different views of multiple stakeholders. The methods used for displaying the different views are color and size. Three types of inconsistent requirements can be found, when one stakeholder has incomplete requirement while other stakeholder has more complete requirement, fully overlapping requirements, and totally disjointed requirements. The conflict resolving is done through agent communication protocol like JADE with ACL.

Mairiza and Zowghi [5] proposed an ontological framework (sureCM) to manage the conflicts between security and usability requirements. The output of the system are lists of conflicts, nature of the conflict based on the impact of the conflicts against different components in software development, and conflict resolution strategy.

Butt et al. [2] proposed a Mandatory, Essential and Optional (MEO-strategy) for requirement conflict resolution. The strategy defined three types of requirements: mandatory requirements, essential requirements and optional requirements.

The output of the framework is a requirement matrix contains the conflicting requirements if any and the suggested solution time. Prevention for mandatory, detection and removal for essential and containment for optional requirements. A case study result shows that the users' acceptance test for system performance, quality and conformity to user needs was achieved successfully.

Mairiza et al. [11] applied an experimental approach to design a framework that manage the relative conflicts among NFRs. A suitable exterminate is designed to apply the metric and measure of the NFRs with the functionality of the system and how to implement the functionality (operationalization). The result of the experiment is the satisfaction level of NFRs in the system. A two dimensional conflict relationship graph is created to determine if there is a conflict between the two NFRs and the severity of any existing conflicts, means is it a strong or weak conflicts depend on the shape of the graph.

Moreover, Mairiza et al. [24] proposed a novel idea of utilizing TOPIS (Technique for Order of Preference by Similarity to Ideal Solution) to resolve nonfunctional requirements conflicts. TOPIS is a goal-based technique for finding the alternative that is nearest to the ideal solution.

The framework takes a two-dimensional graph that shows the relationship between two NFRs. Then, a decision matrix is constructed based on the graph. The technique calculate the distance to each alternative to the ideal solution and choose the closed one, it is the solution that maximize both NFRs. Alebrahim et al. [25] presented a structural method to detect candidate requirements interaction between functional requirements. The proposed method consists of three phases. The first phase is to remove any conflicts after analyzing problem diagrams. In the second phase, the set of candidates conflict requirements are reduced using the information if requirements have to be accomplished in parallel or not. In last phase, the candidates conflict set are reduced by checking if combination of their precondition is fulfilled. A real life example was studied and the results show that the number of possible interactions was decreased and thus, the time for looking into requirements interactions decrease by 95%. The precision was 33% and a perfect recall with 100%.

2) Automatic: The word 'automatic' intended using some tools to analyze and detect the requirements conflicts instead of doing that manually.

Egyed and Grunbacher [26] used an automated traceability techniques to eliminate false conflicts and cooperation. The approach automatically analyzing the requirements to identify requirements that conflict based on their attributes, attributes might be indifferent to one another, cooperative or conflicting. Then, the trace analyzer automatically identifies the trace dependencies among the requirements. Based on the knowledge of trace dependencies, the system can determine to what extent the requirements are overlapping. If two requirements overlap, then the two requirement are conflicts. Whereas if there is no overlap between them, they can't be conflicts.

Kim et al. [9] presented a systematic process to detect and manage requirement conflicts based on requirements partition in natural language. A supporting tool (RECOMA) has been built and two types of conflicts are defined, source conflict and activity conflict. The requirement conflict detection is done through two steps. First, a syntactic method automatically identifies the candidate conflict requirements. Then, the semantic method is used to find the actual requirements conflicts through questions list. By automated a syntactic analysis, the number of requirements to be semantically comparison are reduced. Two cases studies are presented and the results demonstrate that comparison requirements dramatically reduced, and thus the time and effort are decreased.

Kamalrudin et al. [8] explained how to use tractability approach to manage the consistency between textual requirements, abstract interactions and Essential Use Cases (EUCs). An automated tracing tool (Marama AI) is built to help users extract abstract interaction from the textual requirements, mapping the type of interaction and creating the EUC model. It supports traceability and inconsistency checking between the three forms. An experiment results show that 94% of the participants were agree that it is useful and all were say it is user friendly and easy to use.

Moser et al.[13], [18] proposed an automatic semantic based approach for requirements conflict detection. The proposed solution consists of two main phases. First step is to link requirements written in natural language to semantic concepts to build the project ontology. Then the requirements will automatically and semantically analyzed to identify possible conflicts using sets of assertions that should be true for all existing facts.

They defined three types of conflicts that could be detected: conflict between a requirement and a constraint (CRC), conflict between a requirement and a guideline (CRG) and conflict between requirements (CRR).The evaluation results show that the prototype tool (OntRep) found all conflicts while manual conflict analysis found 30% - 80% of the conflicts. Also, the correctness of the proposed approach is 100% compared to 58.8 of false positive in manual analysis.

While Urbieta et al. [19], [20] proposed a model-driven approach to detect requirement conflicts in Web applications in early stage of software development. The approach starts automatically listing the candidate structural and navigational conflicts by structural analysis using the Navigational Development Techniques (NDT) model. Then semantic analysis on requirements is formalized using Domain Specific Language (DSL) for candidate conflicts to avoid false positives which are conflicts that are actually not in conflict.

Resolving conflicts will be done manually using the proposed conciliation rules or by stakeholders' negations. Compared to manual approach, the evaluation shows that system detects 100% of inconsistencies and the time is reduced by 78% which saves 44% of budget.

Nguyen et al. [27] proposed Knowledge Based Requirements Engineering (KBRE) framework. The domain knowledge and semantics of requirements are centralized using ontology and the requirements goal graph is used to detect requirements inconsistencies and overlaps. The explanation for each detected requirements is provided automatically. The case study shows the performance of the system is satisfactory by calculating the running time to detect inconsistencies and precision of detecting inconsistent requirements.

Chentouf [21] presented a solution to OAM&P (Operation, Administration, Management and Provisioning) requirements conflicts.. The proposed method used an Extended Backus-Naur Form (EBNF) as representation language for requirements. The system automatically validates each requirement statement based on validation rules. Then it compares every pair of requirement to detect conflicts according to the seven conflict inference rules. Seven types of conflicts were defined and a proposed solution for each type was presented. To test the scalability, results show that the proposed solution gives an acceptable computation time less than a minute for more than 10,1000 requirements. Also, it scales very well as the number of requirements increase.

3) General Framework: Some works can't be classified as manual or automatic techniques. Thus, they are only considered as general frameworks to detect the conflicts between requirements.

Shehatam et al. [22] proposed a three-level interaction detection framework (DRI-3). Level-1 uses informal approaches to detect accurate and domain known interaction with the help of experts, Level-2 Identifies requirements interaction using semi-formal, semi-formal means systematic steps without formalized methods. Level-3 applies formal approaches to detect

accurate interaction.

Additionally, the paper presented a set of guidelines describing which techniques from (DRI-3) can be used based on the values of different attributes of project. A case study is carried to evaluate the efficiency of using the model comparing to experts without applying the model. The results show that the number of comparison requirements is decreased by 18%.

Mairiza and Zowghi [28] demonstrated the results of the investigation of research on NFRs conflicts that resulted in a catalogue of conflicts among NFRs. The catalogue is a two-dimensional matrix that represents the interrelationships among twenty types of NFRs.

It shows three categories of relative conflicts between the NFRs, absolute conflicts for NFRs that are always in conflict, relative conflict for pair of NFRs that are sometimes conflicted and not conflict for NFRs that never conflict in the literature of NFRs conflict studies.

B. Comparing between Exciting Works

This section analyzes and summarizes the comparison between different techniques to provide a general and quick review on the works done in this area.

To offer better understanding and analysis of existing techniques, they will be classified into different categories as shown in figure 1, categorization is based as follows:

- The first classification is based on the conflict identification method, whether it is done manually by the requirement engineers or automatically using software tools. A class of general frameworks is added to classify some works that detect conflicts without using special techniques.
- The second classification is focused on the type of requirements that the technique will be applied to: functional or nonfunctional requirements.
- The third classification is to determine the scope of the proposed approach to examine if it covers the detection problem, detection and analysis of the conflicts requirements to organize them into different conflict types, and if the proposed approach offers a resolving technique.
- The last classification is based on the representation type for requirements used. If the detection technique uses a specific formalization form, it structures the requirements in a particular model, or it uses an ontology.

Table I summarizes previous works listed by reference number for the research, conflict analysis approach used in identifying conflicts, and category of proposed methods (manually, automatic or just a general framework). It also states the type of requirements the technique is applied to and what is the scope of the technique (i.e. identify, analyze, resolve). It also determines what representation was used to complete or facilitate the technique (formalization, structure model or ontology). The last column indicates whether the proposed technique was supported by evaluation or not. For example, the first row corresponds to a manual technique that

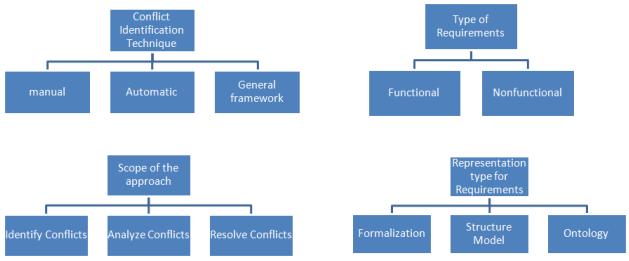

Fig. 1. Categorization of existing techniques for requirements conflicts

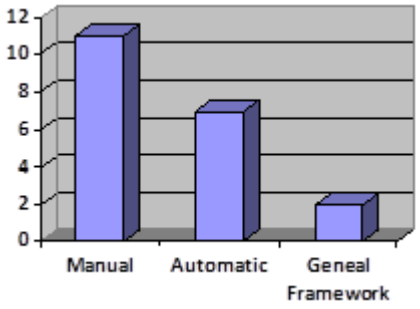

Fig. 2. Analysis results based on identification method

uses a heuristic algorithm to detect functional requirements conflicts and uses schematic versions as formalization form for requirements representation.

The previous works that were studied include 20 papers; these papers are the ones that have a close association with the problem of requirement conflicts. They have proposed different approaches for conflict analysis and detection. For automatic techniques, the conflict analysis approach can be classified into the following four groups: (1) semantic approach for technique that use ontology like [13], [18]; (2) syntax approach when a syntax analysis is done for requirement specification like [9]; (3) graphical analysis when a specific model used like [20], [19], [27]; and (4) tractability approach when tractability technique is used like [26], [8].

The main classification shows that twelve of the works (more than half of them) are manual techniques performed by software engineers whereas only seven are automated tools that help them to find conflicts in requirements, see figure 2.

The analysis of the works as shown in figure 3 demonstrates that eleven of the twenty works work on functional requirements and only six for nonfunctional requirements while there are three works for both of them.

The scope of the proposed solutions was different. As figure 4 shown, almost all research focus on the problem of identifying conflicts, while thirteen of them analyze the conflicts to give different classifications for the conflicts. Only five research studies give guidelines and proposed resolving approaches.

Most techniques used different representation for the requirements to help in analysis and identification of conflicts. Figure 5 indicates that the representation methods used can be divided to three types: either using ontology if the technique using semantic analysis for the requirements like [5], [27],

TABLE I. COMPARISON BETWEEN EXISTING WORKS IN REQUIREMENTS

Reference	Conflict Analysis Approach	Conflict Identification Method	Type of Requirements	Scope of the Approach	Requirement Representation	Evaluation
[3]	Heuristic algorithm	Manual	Functional	Identify	Formalization (schematic versions)	
[6]	Root requirements analysis	Manual	Functional Nonfunctional	Identify Analyze(different degree of conflicts)	Formalization (structure requirement)	✔
[15]	Nonfunctional decomposition model(NFD)	Manual	Nonfunctional	Identify Analyze(grouping conflicts, in-group conflicts) Resolve		
[16]	Integrated analysis of FRs and NFRs	Manual	Nonfunctional	Identify Analyze(mutually exclusive, partial)	Formalization (two canonical form are developed)	
[29]	Model based in UML activity diagram	Manual	Functional	Identify Analyze (7 types of conflicts)	Structure model (Activity Diagram)	
[17]	Non-mathematical technique	Manual	Functional	Identify Analyze (3 types) Resolve	Formalization (semi-formal ontology driven domain-special requirement language)	
[5]	Ontological framework	Manual	Nonfunctional (security and usability)	Identify Analyze (natural of conflict) Resolve	Ontology	
[2]	MEO-strategy	Manual	Functional Nonfunctional	Identify Analyze (mandatory, essential, optional)		✔
[11]	Experimental approach using NFRs metrics and measures as parameters	Manual	Nonfunctional	Identify Analyze (strong, weak)		
[24]	A goal-based technique (TOPIS)	Manual	Nonfunctional	Resolve		
[30]	Graphical method using problem diagram	Manual	Functional	Identify	Structure model (problem diagram)	✔
[26]	Traceability approach	Automatic	Functional Nonfunctional	Identify		
[9]	Requirements partition in natural language	Automatic	Functional	Identify Analyze (source conflicts, activity conflict)	Formalization	✔
[8]	Tractability approach	Automatic	Functional	Identify	Structure model (EUC)	✔
[13],[18]	Semantic based approach	Automatic	Functional	Identify Analyze (CRC,CRG,CRR)	Ontology	✔
[20],[19]	Graphical method using NDT meta model	Automatic	Functional	Identify Analyze	Formalization(DSL) Structure model(NDT requirement meta model)	✔
[27]	Graphical method using requirement goal graph	Automatic	Functional	Identify Analyze	Ontology Formalization(OWL) Structure model (goal graph)	✔
[21]	Validation rules	Automatic	Functional	Identify Analyze (7 types) Resolving for each type	Formalization	✔
[22]	Three-level interaction detection framework	General framework	Functional	Identify		✔
[28]	Investigation of research on NFRs and build a catalogue of NFRs conflicts	General framework	Nonfunctional	Analyze (absolute, relative, no conflict)		

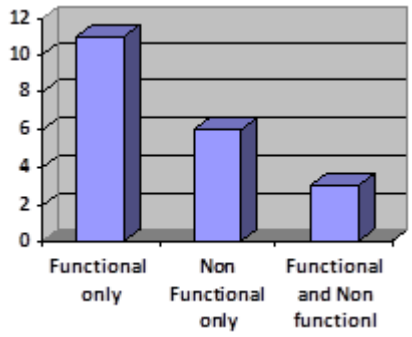

Fig. 3. Analysis results based on type of requirements

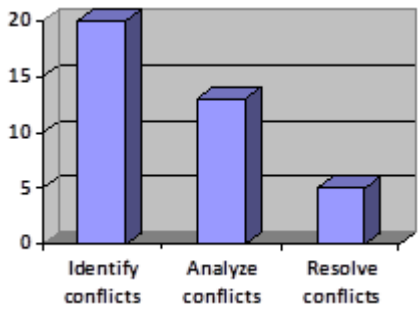

Fig. 4. Analysis results based on scope of the approach

[13], [18]; structural model if graphical analysis is used like [20], [19], [8], [29]; or formalization methods which are different from schematic version in [3], structure requirements in [6], two canonical forms in [16], semi-formal ontology driven domain-special requirement language in [17], DSL in

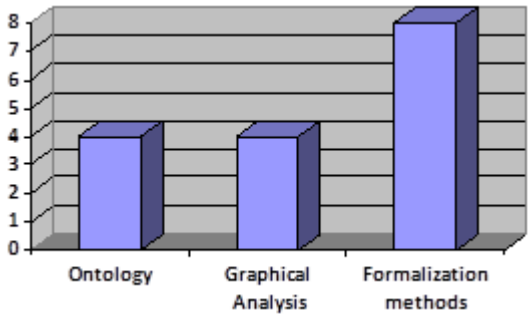

Fig. 5. Analysis results based on requirements representation used

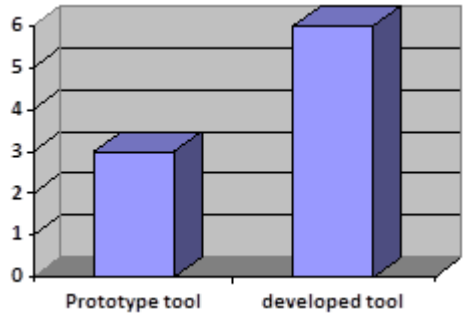

Fig. 6. Analysis results based on type of tool used

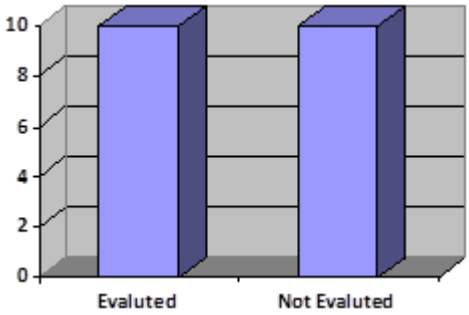

Fig. 7. Analysis results based on if the work is evaluated or not

[20] and [19], OWL in [27], and EBNF in [21]. While some proposed techniques used more than one type of representation like [20], [19] which used ontology and DSL as formalization method, [27] used ontology, OWL as formalization method and goal graph as a structural model.

The tools used in the proposed approaches are either prototype tools like [13], [18], [20], [19], [27]; or developed tools like [26], [9], [8], [21]. [26], [9] developed tracing analysis tools because the conflict analysis approach used is a tractability approach. See figure 6.

Analysis on previous works illustrate that only half of the works were evaluated to test the effectiveness of the proposed method, as shown in figure 7.

According to the importance of evaluation, more analysis was conducted on the evaluated works. Table II summarizes the evaluated works listed by reference number. The second column presents evaluation data that was used to test the system. The literature illustrates that most works use case studies to test the effectiveness of the proposed method; except for when in two works, one uses a survey and the second uses an experiment. The third column explains the goal of the evaluation. The objectives were differed between the following: assessment of the utility, measuring users' satisfaction, evaluation of the effectiveness, demonstration of the tool feasibility, testing of the usefulness and ease of use, and testing of the completeness and consistence of the proposed method.

The fourth column explains the method used in the evaluation. It is clear that almost all works used the comparison in the number of detected conflicts, the validity of the detected conflicts, and the cost when using the proposed method without using any specific approach. Finally, the evaluation results are presented in the last column.

IV. Discussion

The previous sections have discussed in detail the existing works for detection and managing requirements conflicts. However, the topic is still active for researchers in the field of requirements engineering.

The problem of requirements conflicts can be divided into two main sections: identifying the conflicts in requirements; and resolving them. This paper focused on identifying requirements conflicts. The literature review demonstrated that most techniques that are proposed to decrease risks caused by requirements conflicts are manual techniques while the automated approaches are tools based on human analysis. That may incur costs to the project due to human error and wrong decision making.

There are still many gaps in the previous works in identifying requirements conflicts. Detecting conflicts manually takes a long time and effort which may cause delays in the project. In addition, it is fallible since it is done by human effort. Some conflict techniques have tried to automate the detection process by using or building specific tools. Applying some automation to the process would decrease the human effort and time. However, all the automation approaches are still based on human analysis to detect and resolve conflicts. Also, most techniques are proposed techniques that are not evaluated for their efficiency in detecting and resolving conflicts.

There are some important issues that should be taken into consideration when working to find practical techniques for detecting conflict between requirements. First, define what requirement conflict exactly means and what it includes to find a suitable technique to catch the conflict. Then, determine the type of requirements that the technique will work on and which representation method for requirement specification is the most suitable for use. Also, determine when the technique can be applied and in which phase of software development. As final step, determine how to measure the efficiency of the proposed technique.

V. Conclusion

Requirements engineering is a critical part of software development that plays an important role in the software project success. However, there are different issues that may

TABLE II. COMPARISON BETWEEN EVALUATED WORKS

Reference	Evaluation Data	Evaluation Goal	Evaluation Method	Evaluation Results
[6]	Case study: established requirements engineering problem case (Distributed Meeting Scheduler).	Assess the utility.	Compare the number of conflict detection using root analysis and without using it.	Using root analysis technique detects 72 conflicts, while without using it only 9 conflicts is detected.
[2]	Case study: applied MEO-strategy during build (Hostel Management System) as part of university management system.	Measure users' satisfaction.	Conduct a feedback workshop to collect user's feedback.	Users' acceptance test for system performance, quality and conformity to user needs was achieved successfully.
[30]	Case study: proposed approach was used in real life example in domain of (Smart Grids).	Validate the proposed approach.	Compare the number of possible requirements interaction using problem diagram and without using it to measure the effort (time) need to detect the interaction. Measure the precision and the perfect recall using problem domain approach.	The number of possible interactions was decreased and thus, the time for looking into requirements interactions decrease by 95%. The precision was 33% and a perfect recall with 100%.
[9]	Case study: The proposed approach was applied in (Home Integration System (HIS) and (Cellular phone domain).	Demonstrate the tool feasibility.	Compare the total number of comparison to find conflict using the proposed automated tool with the manual approach by developers. Also, compare the time and cost using the two approaches.	The number of comparison using manual approach in His is 378 and in cellular phone case is 666. Comparing to 79 and 100 using the automated proposed approach. While the number of comparison is decrees, the time and cost will decrees.
[8]	Survey: with 8 software engineering post-graduate students.	Test the usefulness and ease of use.	Use Likert scale with five part scale to evaluate the usefulness and ease of use of the proposed approach.	Results show that 94% of the participants were agree that it is useful and all were say it is user friendly and easy to use.
[13],[18]	Case study: real-world industrial case study with 6 project managers and requirement expert.	Evaluate the effectiveness	Compare the number of conflicts detected using the proposed method with the manual approach. Also, compare the percentage of correctness in the two approaches.	The prototype tool (OntRep) found all conflicts while manual conflict analysis found 30% - 80% of the conflicts. Also, the correctness of the proposed approach is 100% compared to 58.8 of false positive in manual analysis.
[20],[19]	Experiment: simulation in real environment of Mosaico.	Measure the efficiency and effectiveness.	Calculate the number of inconsistencies detected and the time and the cost is compared to manual approach.	The evaluation shows that system detects 100% of inconsistencies and the time is reduced by 78% which saves 44% of budget.
[27]	Case study: on traveler social networking system.	Evaluate the effectiveness.	Measure the performance of the system in the number and the precision of detecting inconsistencies.	The performance of the system is satisfactory by calculating the running time to detect inconsistencies and precision of detecting inconsistent requirements.
[21]	Proof-of-concept example , simulation test	Test the completeness and consistence To test the scalability,	Use proof-of-concept Compute the computational time for different number of scalability.	The acceptable computation time less than a minute for more than 10,1000 requirements. Also, it scales very well as the number of requirements increase.
[22]	A case study : smart homes domain	Evaluate the efficiency	Compare the number of comparison done by expert if applying the approach without applying it.	The number of comparison requirements is decreased by 18%.

be caused by giving incorrect requirements and therefore, this results in project failure, which is one of the problems in requirements conflict.

The paper provided a literature review on requirements conflict research and analyzed them to show the limitations and gap in previous works. Also, a more detailed analysis was conducted on the works that were evaluated to illustrate the evaluation methods and data used in the previous works. The literature review demonstrated that most techniques that are proposed to decrease the risks and detect requirements conflicts are manual techniques while the automated approaches are tools based on human analysis. That may incur costs to the project due to human error and wrong decision making. Moreover, most of the proposed approaches were not evaluated to measure their efficiency. At the end, important issues were given as general recommendations when proposing requirements conflicts technique.

REFERENCES

[1] D. Mairiza, D. Zowghi, and N. Nurmuliani, *Managing conflicts among non-functional requirements*, University of Technology, Sydney, 2009.

[2] W. H. Butt, S. Amjad, and F. Azam, *Requirement Conflicts Resolution: Using Requirement Filtering and Analysis* in Computational Science and Its Applications - ICCSA 2011, B. Murgante, O. Gervasi, A. Iglesias, D. Taniar, and B. O. Apduhan, Eds. Springer Berlin Heidelberg, 2011, pp. 383-397.

[3] M. Heisel and J. Souquires, *A Heuristic Algorithm to Detect Feature Interactions in Requirements* in Language Constructs for Describing Features, S. G. Bs. (Hons) and M. R. B. MA, Eds. Springer London, 2001, pp. 143-162.

[4] M. Ramzan, *Intelligent Requirement Prioritization using Fuzzy Logic*, Ph.D., National University of Computer and Emerging sciences, Pakistan, 2010.

[5] D. Mairiza and D. Zowghi, *An ontological framework to manage the relative conflicts between security and usability requirements* in 2010 Third International Workshop on Managing Requirements Knowledge (MARK), 2010, pp. 1-6.

[6] W. N. Robinson, *Surfacing Requirements Interactions* in Perspectives on Software Requirements, J. C. S. do P. Leite and J. H. Doorn, Eds. Springer US, 2004, pp. 69-90.

[7] B. Schr, *Requirements Engineering Process HERMES 5 and SCRUM*, University of Applied Sciences and Arts, Northwestern Switzerland, 2015.

[8] M. Kamalrudin, J. Grundy, and J. Hosking, *Managing Consistency between Textual Requirements, Abstract Interactions and Essential Use Cases* in Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual, 2010, pp. 327-336.

[9] M. Kim, S. Park, V. Sugumaran, and H. Yang, *Managing requirements conflicts in software product lines: A goal and scenario based approach*, Data Knowl. Eng., vol. 61, no. 3, pp. 417-432, Jun. 2007.

[10] W. N. Robinson, S. D. Pawlowski, and V. Volkov, *Requirements Interaction Management*, ACM Comput Surv, vol. 35, no. 2, pp. 132-190, Jun. 2003.

[11] D. Mairiza, D. Zowghi, and V. Gervasi, *Conflict characterization and Analysis of Non Functional Requirements: An experimental approach*, in 2013 IEEE 12th International Conference on Intelligent Software Methodologies, Tools and Techniques (SoMeT), 2013, pp. 83-91.

[12] Hausmann, Jan Hendrik and Heckel, Reiko and Taentzer, Gabi, *Detection of conflicting functional requirements in a use case-driven approach: a static analysis technique based on graph transformation* in Proceedings of the 24th international conference on software engineering, ACM, 2002, pp. 105-115.

[13] T. Moser, D. Winkler, M. Heindl, and S. Biffl, *Requirements Management with Semantic Technology: An Empirical Study on Automated Requirements Categorization and Conflict Analysis* in Advanced Information Systems Engineering, H. Mouratidis and C. Rolland, Eds. Springer Berlin Heidelberg, 2011, pp. 3-17.

[14] M. Shehata, A. Eberlein, and A. Fapojuwo, *IRIS: a semi-formal approach for detecting requirements interactions* in Engineering of Computer-Based Systems, 2004. Proceedings. 11th IEEE International Conference and Workshop on the, 2004, pp. 273-281.

[15] E. R. Poort and P. H. N. de With, *Resolving requirement conflicts through non-functional decomposition* in Fourth Working IEEE/IFIP Conference on Software Architecture, 2004. WICSA 2004. Proceedings, 2004, pp. 145-154.

[16] V. Sadana and X. F. Liu, *Analysis of Conflicts among Non-Functional Requirements Using Integrated Analysis of Functional and Non-Functional Requirements* in Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International, 2007, vol. 1, pp. 215-218.

[17] T. L. Heng and L. T. Ming, *Using multi-coordinated views with agent communication protocol to detect and resolve inconsistent requirements to improve accuracy* in Information Technology (ITSim), 2010 International Symposium in, 2010, vol. 2, pp. 1041-1044.

[18] T. Moser, D. Winkler, M. Heindl, and S. Biffl, *Automating the detection of complex semantic conflicts between software requirements* in The 23rd International Conference on Software Engineering and Knowledge Engineering, Miami, 2011.

[19] M. J. Escalona, M. Urbieta, G. Rossi, J. A. Garcia-Garcia, and E. R. Luna, *Detecting Web requirements conflicts and inconsistencies under a model-based perspective*, J. Syst. Softw., vol. 86, no. 12, pp. 3024-3038, Dec. 2013.

[20] M. Urbieta, M. J. Escalona, E. R. Luna, and G. Rossi, *Detecting Conflicts and Inconsistencies in Web Application Requirements* in Current Trends in Web Engineering, A. Harth and N. Koch, Eds. Springer Berlin Heidelberg, 2012, pp. 278-288.

[21] Z. Chentouf, *Managing OAM&P requirement conflicts*, J. King Saud Univ. - Comput. Inf. Sci., vol. 26, no. 3, pp. 296-307, Sep. 2014.

[22] M. Shehata, L. Jiang, and A. Eberlein, *A requirements interaction detection process guide* in Canadian Conference on Electrical and Computer Engineering, 2004, 2004, vol. 3, pp. 1753-1756 Vol.3.

[23] A. Sardinha, R. Chitchyan, J. Arajo, A. Moreira, and A. Rashid, *Conflict Identification with EA-Analyzer* in Aspect-Oriented Requirements Engineering, A. Moreira, R. Chitchyan, J. Arajo, and A. Rashid, Eds. Springer Berlin Heidelberg, 2013, pp. 209-224.

[24] D. Mairiza, D. Zowghi, and V. Gervasi, *Utilizing TOPSIS: A Multi Criteria Decision Analysis Technique for Non-Functional Requirements Conflicts* in Requirements Engineering, D. Zowghi and Z. Jin, Eds. Springer Berlin Heidelberg, 2014, pp. 31-44.

[25] Alebrahim, Azadeh and Faßbender, Stephan and Heisel, Maritta and Meis, Rene, *Problem-based requirements interaction analysis*.

[26] A. Egyed and P. Grunbacher, *Identifying requirements conflicts and cooperation: how quality attributes and automated traceability can help*, IEEE Softw., vol. 21, no. 6, pp. 50-58, Nov. 2004.

[27] T. H. Nguyen, B. Q. Vo, M. Lumpe, and J. Grundy, *KBRE: a framework for knowledge-based requirements engineering*, Softw. Qual. J., vol. 22, no. 1, pp. 87-119, Apr. 2013.

[28] D. Mairiza and D. Zowghi, *Constructing a Catalogue of Conflicts among Non-functional Requirements* in Evaluation of Novel Approaches to Software Engineering, L. A. Maciaszek and P. Loucopoulos, Eds. Springer Berlin Heidelberg, 2011, pp. 31-44.

[29] C.-L. Liu, *Ontology-Based Requirements Conflicts Analysis in Activity Diagrams* in Computational Science and Its Applications - ICCSA 2009, O. Gervasi, D. Taniar, B. Murgante, A. Lagan, Y. Mun, and M. L. Gavrilova, Eds. Springer Berlin Heidelberg, 2009, pp. 1-12.

[30] A. Alebrahim, S. Fabender, M. Heisel, and R. Meis, *Problem-Based Requirements Interaction Analysis* in Requirements Engineering: Foundation for Software Quality, C. Salinesi and I. van de Weerd, Eds. Springer International Publishing, 2014, pp. 200-215.

[31] *How to Deal with Stakeholders Conflicts in Requirements Gathering?*, [Online]. Available: https://www.researchgate.net/post/How_to_deal_with_stakeholders_conflicts_in_requirements_gathering2. [Accessed: 20-Apr-2016].

[32] *Scope - How Do You Manage Conflicting Stakeholder Demands? - Project Management Stack Exchange*. [Online]. Available: http://pm.stackexchange.com/questions/1399/how-do-you-manage-conflicting-stakeholder-demands. [Accessed: 20-Apr-2016].

[33] *I Have Two Business Stakeholders Who Have Conflicting Requirements - Projectconnections.com*. [Online]. Available: http://www.projectconnections.com/knowhow/burning-questions/resolving-conflicting-stakeholder-requirements.html. [Accessed: 20-Apr-2016].

[34] *Conflicting Requirements*. [Online]. Available: http://c2.com/cgi/wiki?ConflictingRequirements. [Accessed: 20-Apr-2016]