Soft Comput (2018) 22:6645–6656
https://doi.org/10.1007/s00500-017-2769-2

FOCUS

Assessing content-based conformance between software R&D documents and design guidelines using relevance links

Jong-Hwan Shin[1] · **Du-San Baek**[1] · **Byeongjeong Lee**[2] · **Jung-Won Lee**[1]

Published online: 9 August 2017

Abstract Research-oriented software groups are groups that carry out research on original technology for software. They also develop prototype software to demonstrate the proof of concepts of their research. The prototype software gets more complex and grows rapidly along with the growth of software technologies. Software documents describing the characteristics of the software are also diverse, and their contents are becoming enormous. However, the groups lack proper documentation on their research and software. The main reason for this problem is that research documents are not well managed, even though they have significant influence on this group and their software. Therefore, a systematic approach for managing the generated documents is required on software R&D process. We propose a method that can reduce poor documentation related to software research and development. We construct design guideline models based on the best practices and represent each measurement of design guideline models by queries of semantics-aware traceability links. Then, we use a semi-automated method to assure the conformance of R&D documents to the guidelines. Finally, we provide an explanatory guideline for assessment results to explain the detail of the assessment and provide the advice for the quality improvement in the evaluated documents. We evaluated the documents generated from our previous R&D project to show the applicability of our method. As a result, our method can help improve the quality of software R&D project documentation in a short time.

Keywords Conformance assurance · Relevance link information model (RLIM) · Traceability · Software documentation quality · Expert assessment · Design guideline

Communicated by G. Yi.

✉ Jung-Won Lee
jungwony@ajou.ac.kr

Jong-Hwan Shin
sjh1334@ajou.ac.kr

Du-San Baek
whitedusan@gmail.com

Byeongjeong Lee
bjlee@uos.ac.kr

[1] Department of Electrical and Computer Engineering, Ajou University, 338 Wonchun-Hall, 206 Worldcup-ro, Yeongtong-gu, Suwon, Korea

[2] Department of Computer Science and Engineering, University of Seoul, Seoul, Korea

1 Introduction

Research-oriented software groups consist of researchers who study original technologies for software, such as an algorithm, a program, and a framework, to deal with the problem of current software technology. They need to develop software that contains their original technology for demonstrating the effectiveness of their studies. In short, a software project in research-oriented software group has two phases: research phase and development phase. Since computers and software are becoming essential in every field, the problems solved by software are becoming more complex and diverse. Therefore, studies for solving the software problems get complicated continuously. Software created by the groups also has grown significantly in the last few decades in terms of complexity and size. Consequently, documentation that explains researchers, requirements, design, and other features of software becomes large and more complex.

However, software projects aiming for original technologies generally experience poor documentation. There are several reasons for the documentation; however, the root cause of this problem is the lack of systemic management and maintenance of documents created during research phases. Two types of documentation are generated during a project of a research-oriented software group, namely research documents and development documents. Development documents such as software requirement specification and software design description are documents for implementation of software. They reflect various aspects of software, such as requirements, designs, and testing of software. These documents have various systemic ways for creating and maintaining documents. The contents in these documents are linked to each other and managed with integrity; therefore, they can reflect the characteristics of the software very well.

On the contrary, the research documents are not generally maintained in an organized way. Researchers in research-oriented software groups create and maintain research documents such as research plans, annual reports, and technical reports for the research phase of projects. These types of documents are necessary to the R&D life cycle because they play important roles in planning and describing research process. Although the importance of research documents on software R&D projects for original technology is significant, current software configuration management schemes mainly focus on software development documents that describe requirements, design, and testing of software. Therefore, the current techniques cannot consider on research documents in spite of the importance of research documents. Poor documentation results and leads to an increase in the technical debt of the project.

Since the influence of research phase in research-oriented software group to its software is significant, the problem with research documents can result technical debt in various ways, including the lack of time in development. Therefore, we considered methods for research documentation in the life cycle of software configuration management and the methods for improving the quality of research documents.

Various guidelines including the measures for management of software development documents require ensuring traceability of overall documents. Traceability can be defined as explicit links or mapping that are generated as a result of transformations (Aizenbud-Reshef et al. 2006). Basically, it does not provide semantics for relationship. This type of traceability link has a limitation in understanding the exact meaning between items because it has limits on the expression of various associations between items. Therefore, it is difficult to evaluate the quality measurement and improvement in large collection of documents with this type of traceability. Thus, several researches have been conducted so that traceability links have various meanings for many years. However, traceability links have been mainly focused on development documentation, especially targets requirement traceability where relation can be limited. However, it was not suitable for research documents that the relations between items can be various.

Therefore, we proposed a relevance link information model (RLIM) techniques that can present the semantic relationships and the traceability among research documents and development documents in an earlier work (Baek et al. 2016b). Unlike development documents that specify precise requirements, research documents are composed of abstract and general contents such as research objectives. Hence, efficient management of research documents cannot be done with existing software configuration management scheme. Therefore, we proposed relevance link that defines the semantics of the traceability link beyond the limitation of existing traceability techniques and outlined seven semantic and structural rules. Second, we proposed a configuration management method using the RLIM (Baek et al. 2016a). This method uses the RLIM to identify the content types of the documents and maps the corresponding items into predefined semantic rules to generate relevance links. Next, it checks the consistency and completeness of the contents using the generated RLIM. This work is the basis for performing a content-based test of software R&D documents.

On the basis of the studies mentioned above, it is possible to maintain the integrity and traceability of documents by carrying out the configuration management of R&D documents. However, problems, such as missing contents, related to R&D documentation can occur, since researchers may have a lack of training on the software documentation best practices. Therefore, in this paper, we propose a detailed content-based methodology for the assurance of the quality of research documentation along with development documents to address the problems described above. This paper focuses on the automation of conformance testing for a conformance assurance method between the design guideline and software R&D documents (Shin et al. 2016). In this paper, we propose the transformation of metric types to help transformation of metrics for conformance testing and the detail of how conformance testing is conducted.

First, we let researchers generate RLIM with expressing semantic and structural relationship among the type of document contents. Second, we define the structure of design guideline model (DGM) that plays as guidances for maintaining and improving qualities of documents. A design guideline model is made for a document type and consists of a goal model and explanatory guidelines that explain the goal model for user feedback. Next, we create design guideline models by extracting obligations related to the qualities of R&D documents from relevant software engineering standards that reflect the software industries' best practices.

Third, we define an automated evaluation method for checking the qualities of documents. First, instantiation of RLIM and DGM with actual instances of document collection takes place. RLIM and DGM are converted into relevance link information and design guideline, respectively. Then, we transform the design guidelines into relevance links for automatic document evaluation by converting the metrics, which are the evaluation targets and the evaluation criteria, to fit our proposed document test system for automated evaluation. Then, an expert system performs the quality evaluation with relevance link information and design guideline. Finally, the evaluation report is generated, which summarizes the evaluation result and explains the rationale that the user can easily understand.

Our proposed method facilitates the securing and maintenance of the quality of research and development documents as well as ensuring the integrity of contents and traceability between other documents. Our method also can help users to carry out research and development of software in a more efficient way with automated evaluation of documents.

2 Related work

Several standards and works for supporting software documentation exist (Barker 1990; ISO/IEC 2008; Bourque and Fairley 2014). Still, poor documentation is one of the major reasons of technical debt (Allman 2012), and many studies for improving software documentation have been conducted. Some automated approaches for analysis and evaluation of document quality have been proposed (Dautovic et al. 2011; Jain et al. 2009; Thitisathienkul and Prompoon 2015). Besides, some methods that help minimize time-consuming tasks for reviewers during review processes and writers during document writing processes have been reported (Cyra and Gorski 2011; Shen et al. 2013; Antonino et al. 2015).

2.1 Software R&D documentation

Software documentation is defined as follows: 'the design, planning, and implementation of any interface element, and a software system to enhance the system's usability (Barker 1990).' Software documentation is divided into several categories on the basis of software processes. The documents used in a software implementation include a requirements specification that specifies software requirements, a software design specification that describes the design aspects in detail, and a test design and results report for software testing (ISO/IEC 2008; Bourque and Fairley 2014).

Research documents do not belong to the categories of general software documentation defined in software engineering standards. However, these documents are the work product generated at different milestones during the research and development process. The research documents include various types of documents such as research plans and annual reports. Because the regulations related to the research documents are not stringent, the metrics for validating the contents and the qualities of research documents usually differ among project groups. Therefore, we have identified common document types used in various R&D projects and defined formalized contents (Baek et al. 2016b).

Documents used for software research have different aspects from development documents. Research documents are written for researchers' understanding and for documenting research progress. These documents are often specified as high-level requirements such as 'research purpose.' However, documents used for implementation of software focus on realizing the requirements in the requirements elicitation. Therefore, in these documents, content is provided in as much detail as possible to reduce ambiguity. Therefore, some limitations exist in applying existing software configuration management techniques to research documents.

2.2 Automated software documentation evaluation

Several studies have been conducted on automated evaluation of software documentation because analysis and validation of numerous documents require a substantial amount of time. Since the elicitation and specification of the software requirements are the first steps of software development and unresolved requirements cause technical debt of the development phase, many studies have been carried out for ensuring that software requirements documents are written correctly.

Wilson et al. set specification quality attributes of software requirements, such as complete, correct ranked, unambiguous, consistent, modifiable, traceable and verifiable. Authors perform keyword-based analysis and quality measurements of software requirements (Wilson et al. 1997). They set quality indicators that can measure quality attributes from their quality model and determine how often those terms are presented. This work shows the need for quality control of software documents and a simple method to improve document quality, but it cannot directly evaluate high-level quality attributes such as traceability.

Jain et al. (2009) proposed a requirement analysis tool that can analyze and measure the qualities of requirements documents. The tool uses a controlled natural language approach for requirements analysis. The tool performs lexical analysis for conformance of the requirement specification template. Then, it performs semantics analysis for checking the completeness with state machines of each requirement. The tool also generates a helpful message for unfinished specifications. This work demonstrated the reduction in the time on review process. However, this method mainly focuses on the syntax of requirements.

2.3 Helping documentation review

Several studies have been proposed to help software engineers write software documentation efficiently and to reduce the effort of regulators in a regulatory review during regulatory conformance process. This is achieved through the use of templates that reflect the best practice for software documents or by automatically providing the relevant contents when reviewing documents.

Dautovic et al. (2011) proposed an automatic checking method of overall software development documents. This method proposes a visitor pattern to traverse document contents and simple rule-checking mechanism for quality measurement. However, the rules they created concentrated on the structural and formatting aspects of the document.

Cyra and Gorski (2011) defined a Trust Case template that specifies the requirements of the standard and explains how it should be accomplished by the evaluators and the stakeholders of the software, using various data including relevant standards and reference documents. This method first collects development documents and uses them as evidence for the Trust Case template to assess and achieve compliance with certain software standards. The method then provides guidelines for achieving compliance with the standards by providing the Trust Case template to stakeholders of software development. However, in this method, only the conformance to the software development standard is evaluated. Therefore, consideration for supporting research activities is insufficient, and there is a burden to create a trust case with separate tasks from documentation itself.

Shen et al. (2013) proposed a mechanism that utilizes traceability information to help regulatory process of safety critical software. They represent traceability of documentation with a simple linked list instead of a complicated graph. They proposed a traceability mechanism to the regulatory review process. However, they did not consider the evaluation of document quality and did not consider explanatory guidelines.

Antonino et al. (2015) suggested parameterized safety requirements templates for ensuring traceability throughout software documentation in the safety-critical system domain. They used controlled natural language to avoid ambiguity and a safety requirement decomposition pattern, which is a model-based structural guideline, for safety requirements.

In summary, the above methods cannot be used directly to solve the problems we presented. First, automated approaches for document evaluation mainly focus on requirements specification. In addition, these methods also concentrate on the quantitative quality of the documents. Therefore, considering the semantics of documents is restricted. In the case of supporting review and documentation methods, research documents are not considered in the management life cycle.

3 Conformance assurance of research documents

We have adopted the following approaches for assurance of conformance between the design guideline and software research documentation. First, since most of the software standards require traceability, our method adopts a model-driven traceability approach for presenting the relations between document contents. We use semantics-aware traceability for representing research and development documents to relevance link information.

Second, we define design guideline model that becomes criteria for evaluating the qualities of documents. A DGM is composed of a goal model and explanatory guidelines that explain the evaluation criteria to users. Then, we develop design guideline models on the basis of various software engineering standards to provide proper knowledge of the software best practices to our method. In this step, we extract related obligations from related software engineering standards. The extracted requirements are expressed as an expert goal model for the purpose of visualization and direct measurement of software R&D documents. Furthermore, we transform the metrics of the goal model into expressions of semantic and structural rules for automatic evaluation with less user effort.

Third, the evaluation of documents is performed. Instantiation of RLIM and DGM is conducted first. RLIM and DGM are transformed to relevance link information and design guideline that consist of actual contents of the documents on the software R&D project, respectively. The conformance to design guideline is checked in an automated manner using an expert system, which is a rule-based system that can check design guideline on the relevance link information constructed on the first step. However, because the expert system has difficulty in semantic analysis, we check the semantic rules of the relevance links using a separate validation module. After the evaluation, the test system creates a report of results and adds expert assessment guidelines to help users understand the results of the evaluation. Finally, we present a series of explanatory guidelines to explain the criteria of the evaluation to users.

3.1 Relevance link information model

Traceability is defined as the 'ability to establish links between source artifacts and targets' (Aizenbud-Reshef et al. 2006). This attribute is essential for every software engineering standard. However, the existing traceability information model does not support association relations with various meanings because the definition of a relation is confined to requirement-based traceability. A trace link indicates a transitive relationship while research documents have non-requirement contents and nontransitive relations (Baek et al. 2016a). The relations that do not specify its meaning are

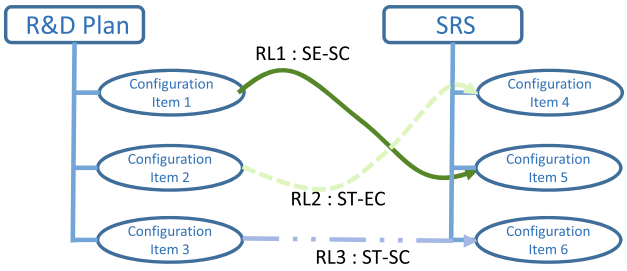

Fig. 1 An example of RLIM

appropriate with development documents which have solid contents for implementation; however, they are inappropriate with research documents because the contents may not be matched directly to each other.

Therefore, we extended the existing trace link to create a relevance link (RL) to overcome above limitations (Baek et al. 2016a). An RL is composed of two principal components: corresponding items that are two configuration items having a relationship to each other and the relevance rule that is the type of the relation between the two configuration items. We defined the total of seven relevance rules to represent structural and semantic relationship between the items on software R&D documents. In other words, a relevance link is a mapping between two corresponding items and a relevance rule.

Figure 1 shows an example of the RLIM with two documents types, namely the R&D plan and the software requirements specification (SRS). Each document has three configuration item types. Three RLs can be found in this example. In RL1, the corresponding items are configuration item 1 on the R&D plan and configuration item 5 on the SRS. Furthermore, the relevance rule of the link is SE-SC, which stands for the semantic sufficient condition. SE in the relevance rule means 'semantic,' that is, the two items are related semantically. Also, the 'SC' stands for the target item which includes the contents of base item. Therefore, the target item includes the contents of the base item semantically. For example, a final goal includes its subgoals semantically.

In RL2, the two configuration item types have an ST–EC relationship. 'ST' means 'structural' and 'EC' means 'efficient' in this case. Therefore, it stands for the structurally efficient condition that the target item includes the base item structurally. For example, a chapter named 'current status of technology' may include recent technology advances and intellectual properties relevant to the technology.

This model can be used for tracking and managing configuration items of research documents. The researcher identifies the document types to be managed as reference document types. Then, the researcher identifies relationships as relevance rules among the configuration items of the reference document types. The relevance link information is constructed in this process. We use this information to evaluate the integrity, quality, and traceability of research and development documents.

3.2 Design guideline model

We propose design guideline model (DGM) that reflects the knowledge of best practices in software development and measures conformance to the knowledge directly. A design guideline model contains some objectives to be achieved, a goal model consisting of questions and metrics, test methods for evaluating metrics, and guidelines for describing the rationale of the test.

We first defined the structure of expert assessment goal model. The goal model has a combined structure of a goal, question, metric (GQM) model (Basili et al. 1994), and the goal structuring notation (GSN) (Kelly and Weaver 2004) for satisfying the expression of the structure of referenced standards and direct measurement of quality metrics.

Figure 2 shows the structure of the goal model for the example of automotive domain. In Fig. 2, Fig. 2a shows the GSN structure of the proposed goal model, and Fig. 2b shows the GQM structure of the proposed goal model. The GSN structure in Fig. 2a mainly focuses on the method of achieving the ultimate goal. In this case, the ultimate goal is the safety assurance of R&D artifacts. Context 1.1 and 1.2 explain the contexts of safety assurance. Strategy 1.1 shows the method of achieving the ultimate goal and providing appropriate descriptions of relevant documents. The justification shape next to Strategy 1.1 justifies the rearrangement of common requirements and the elimination of nonrelated requirements for the context. Goal 2.1 and Goal 2.2 show the subgoals from Strategy 1.1, and the subgoal is providing a proper description of each document presented in Context 1.2., and each subgoal is divided to solutions. For example, Goal 2.1 is divided to four solutions, namely describing safety culture nurturing, safety plan, item description, and hazard analysis and risk assessment.

Figure 2b shows GQM structure of Solution 1.1. Each solution from Fig. 2a has a similar measurement structure to that in Fig. 2b. Figure 2b shows what is the solution to be achieved in the first level. The questions on the second level present the questions to confirm that the solution is achieved. The metrics on the lowest level are metrics to answer the above questions. Setting a proper question is not a trivial task; therefore, we match each question to subsections presented on the relevant standards. Each component of the structure is explained below.

- Ultimate goal: define the ultimate goal that the project has to achieve.
- Strategy: define strategies required for achieving the ultimate goal.

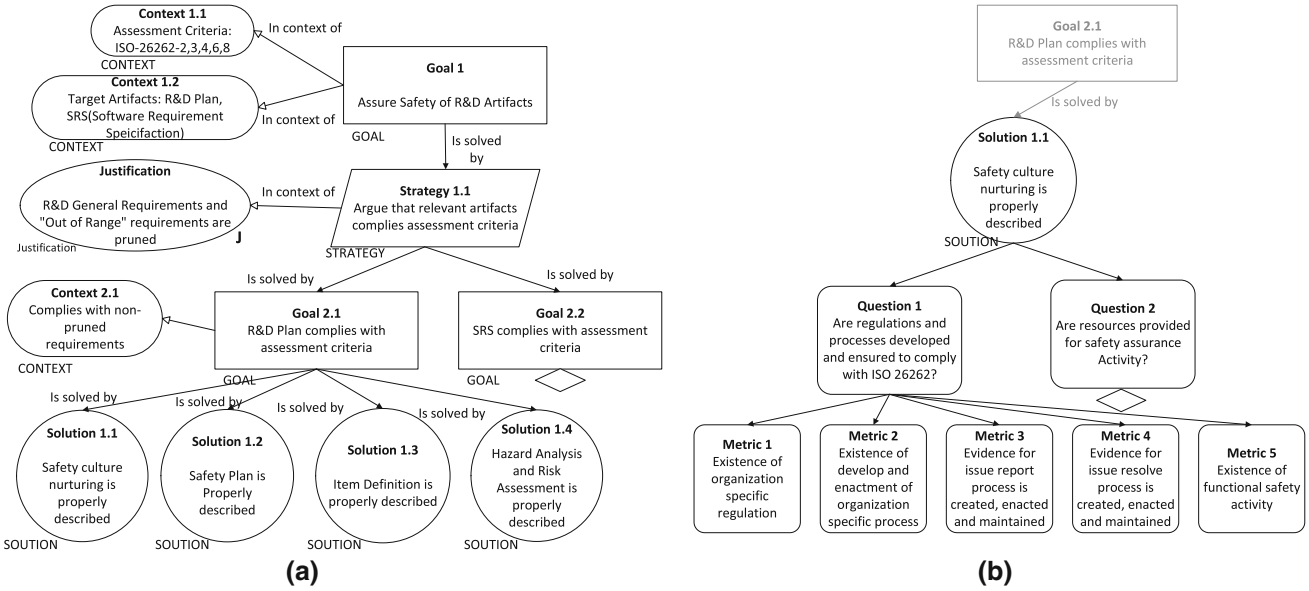

Fig. 2 Proposed goal model **a** GSN structure of proposed goal model, **b** GQM structure of proposed goal model

Table 1 Extracted requirements from relevant software engineering standards

	R&D general	Automotive	Medical device
Target documents	Software R&D plan, software requirements specification		
References	IEC 12207, ISO 26262, IEC 62304, Korean NRF R&D document template	ISO 26262-2,3,6,8	IEC 62304
Extracted requirements	29 Items	28 Items	23 Items

- Justification and context: facilitate the elimination of irrelevant requirements and ensure the legitimacy of the defined goals and strategies.
- Subgoals: define the target documents to be managed.
- Solution: define an appropriate solution for each document.
- Question: define questions that need to be answered when checking whether the solution is obtained.
- Metric: define metrics that need to be measured to answer the question.

Then, we extracted the requirements that the target documents must comply with international software standards in three different domains. This was done by extracting all requirements about the target documents from the relevant guideline first and then removing the requirements not related to the R&D phase. We also rearranged the common requirements on multiple standards to general guideline. The following results were obtained after extraction. Table 1 lists the extracted requirements for software R&D plans and SRSs in three domains: general R&D, automotive, and medical devices domains. We referenced the IEC 12207 standard and document templates provided by Korean National Research Foundation for general software R&D, ISO 26262-2,3 and 26262-8 for automotive software, and IEC 62304 for medical devices (IEC 2006; ISO/IEC 2008; ISO 2011a, b, c, d).

We define the expert evaluation guideline to help users in better understanding of the procedure and the measure of the evaluation. We defined the sentence structure on the basis of the Easy Approaches to Requirement (EARS) template (Mavin et al. 2009) and Rupp's template (Pohl and Rupp 2011) and the result of a software message research (Fagan et al. 2015) to present clear and helpful messages for researchers and developers. Figure 3 shows the structure of the expert guidelines.

In Fig. 3, we explain the source of the metric first. Second, we indicate the degree of the metric being adopted. After that, the corresponding metric name is presented. Further, we present the target document name of the metric. Then, if the metric is valid for a specific situation, we add the condition that the metric is valid. Finally, the goal of the metric is presented. With this structure, the guidelines are automatically generated.

Then, we define the test methods for each metric. The evaluation method of the metric is constructed through the following method for efficient evaluation. First, we clas-

{The source of metric} {requires, recommends} {metric name} at {target document name} on {condition(optional)} for {metric's goal}.

(a)

IEC 12207 6.2.4.2 and Research Project Template for original technology requires recent projects of research manager at Software R&D Plan for proper description organization and task and responsibility assignment.

(b)

Fig. 3 Expert guideline structure and example

sified the types of the metrics as 'Y/N,' 'SEMANTIC' and 'TRACE.' Then, the evaluation ranges of the each metric are classified as shown in Table 2. We referenced the evaluation model shown in Kim et al. (2015) at this stage.

As shown in Table 2, the Y/N metric can be measured simply by checking for the presence of the target item. However, evaluating the semantic metrics is impossible by checking the existence of the target item solely because this type of metrics needs the measure of conformance to its targeted semantics; therefore, the result of semantic analysis on the target element is required.

In this case, a validation module analyzes semantics on the target item. Finally, there is a trace metric that the target element is related to other items and needs to be compared to other items. In this case, we first check the presence of the target content, and then, we check the relevance link. Then, the validation module checks the confidence score of the relevance link's semantic rule.

All metrics are transformed to rules that evaluated automatically. The transformed metrics to rules are defined as a query for certain relevance links or contents of documents. Figure 4 shows some examples of the transformed metrics. There are three metrics presented in this example. Each metric has conformance criteria. We can convert the conformance criterion to a trace query. In the measurement for assessment, RLIM (t1, ci1, t2, ci2, rr) means a relevance link that has corresponding items t1 from ci1 and t2 from ci2 with the relevance rule rr. Ele(x) means the element type of x is Ele. For example, the first guideline has conformance criteria that have no conflict on a trace link between two items; therefore, it checks the presence of two items and relevance link; then, it checks conflict between the links.

Table 2 Metric types and conformance evaluation procedure

Metric type	Description	Evaluation procedure
Y/N	A metric type only requires the existence of certain configuration items	1. Check the existence of the target item
SEMANTIC	A metric type requires the existence and semantic analysis of the target content	1. Check existence of the target item 2. Semantic analysis of the target content by the validation module
TRACE	A metric type requires a relation check between the target item and other configuration items	1. Check the existence of the target item 2. Validate the relevance link of the target item 3. Perform semantic analysis of the relevance link by the validation module

Guideline	Conformance Criteria	Trace Query for assessment (1st order logic)
Internal Consistency of SW Reqs.	If there is a trace link between two items, two items must have no conflict	∃t1∃t2∃rr (Req.(t1)∧Req.(t2)∧Rel.Rule(rr) ∧ RLIM(t1,SRS,t2,SRS,rr) ∧ Conflict(t1, t2))
Integrity between Research Goal and SW Reqs.	Research Goal and at least 'Threshold'% of SW requirements should have semantic sufficient (SE-SC) condition.	∃t1∃t2 (Research Goal(t1) ∧ Req.(t2) ∧ RLIM(t1,SRS,t2,SRS,SE-EC))
Conformity between Research Goal and Plan of Action	Make sure that corresponding items have at least semantic sufficient (SE-SC) condition	∃t1∃t2 (Research Goal(t1) ∧ Plan of Action(t2) ∧ RLIM(t1, Research Plan, t2, Research Plan, SE-SC))

Fig. 4 An example of transformed metrics

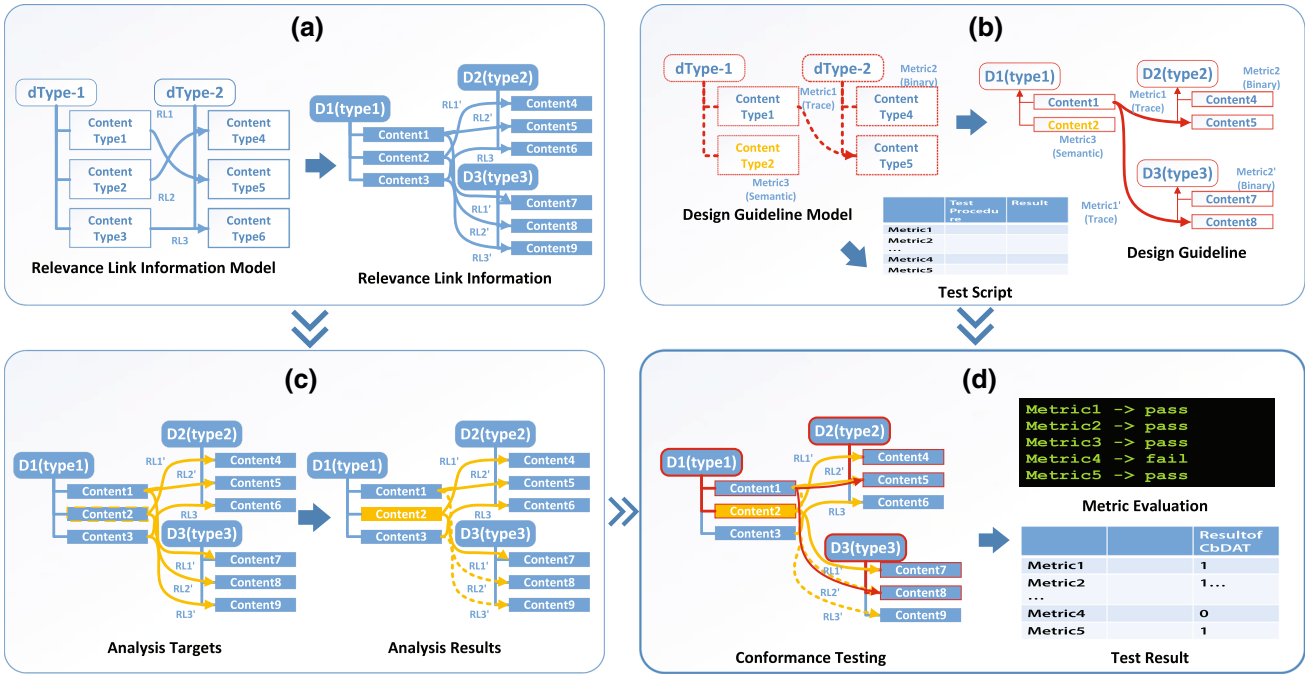

Fig. 5 Procedure for content-based conformance testing **a** RLIM instantiation, **b** DGM instantiation, **c** semantics analysis, **d** content-based document conformance testing

3.3 Content-based conformance testing

A system checks if the target documents comply with DGM on this step. This process requires the contents and RLIM of the target document and DGM. Then, it produces the evaluation report of the test. We define the three components of the testing system as follows.

- Test controller: handles user interaction and the overall system regarding RLIM, DGM instantiation, or test execution.
- Test execution engine: includes an expert system for rule assessment and knowledge base for the expert system.
- Validation module: validates the semantics of required relevance links or document contents.

Figure 5 shows the overall procedure for conformance testing. This process consists of the following four steps. First, the instantiation of RLIM occurs. Then, the DGM instantiation process takes place. Next, the validation module performs semantic analysis. Finally, conformance testing is executed. Through this series of processes, the evaluation report is generated.

In the beginning of the test execution, the RLIM and the DGM of target document types are instantiated to be defined to the actual documents (a)(b). The RLIM and the DGM are instantiated and inserted into the knowledge base on the test execution engine. The knowledge base with disambiguated contents was built on the basis of Sanna et al. (2015).

First, RLIM instantiation is executed (a). This step started with the identification of the current documents from the project context and its contents. The RLIM shows the created model, and this is defined by document type. Since RLIM is generated for each document type, it needs to be redefined for the actual instances on the project. Defined RLs are copied for each instance of the target document types as shown in Fig. 5a. If no instance for a defined RL exists, the RL marked as inappropriate. As a result, relevance link information is created and inserted into the knowledge base of the test execution engine.

Figure 6 shows the procedure of RLIM instantiation. The input is ProjectContext which is the list of documents of the project and the RLIM defined for the document types. the output is relevance link information defined for the ProjectContext. This procedure processes every RLIM defined by user. In a RLIM, it finds the documents from ProjectContext with types of base and target item. Then, it generates RLs for existing document instances of the project.

After that, an instantiation of the DGM is performed as shown in Fig. 5b. A design guideline and its metrics can be selected by considering the project domain, the user's selection, and the progress of the project. The design guideline in Fig. 5b is also defined by the document type and redefined for the project context. Therefore, the context of the project used in Fig. 5a is obtained, and the actual document

Algorithm 1 Instantiate RLIM for ProjectContext

```
Input:
    ProjectContext - the list of documents of the project
    RLIM - the RLIM defined for the document types
Output:
    RLI - the RLI defined for the ProjectContext
    RLINum - the number of created RLI
 1: procedure InstantiateRLIM
 2: begin
 3:   for seq = 0 to sizeof(RLIM) do
 4:     rl ← RLIM(seq)
 5:     dB ← getDocOfType(rl.baseItem)
 6:     dT ← getDocOfType(rl.targetItem)
 7:     for bSeq = 0 to sizeof(baseSeq) do
 8:       for tSeq = 0 to sizeof(targetSeq) do
 9:         RLI[Rseq] ← RLIM(dB[bSeq], rl.ci1, dT[tSeq], rl.ci2, rl.rr)
10:         Rseq ← Rseq + 1
11:   RLINum ← Rseq
12: end
```

Fig. 6 Algorithm of RLIM instantiation

Algorithm 2 Instantiate DGM for ProjectContext

```
Input:
    ProjectContext - the list of documents of the projectd
    DGM - the DGM defined for the document types
Output:
    DG - the DG defined for the ProjectContext
 1: procedure InstantiateDGM
 2: begin
 3:   for each DGM do
 4:     metrics ← getMetricModel(DGM)
 5:     for each metrics do
 6:       Initalize mSeq to 0
 7:       if metric.MetricType = 'TRACE' then
 8:         rlim ← getRLIM(metric.rlim)
 9:         instatiateRLIM
10:         mseq ← mseq + 1
11:       else
12:         dT ← getDocuTypeOf(metric.targetRdTypeId)
13:         cT ← getDocuTypeOf(metric.targetConfItemId)
14:         for seq = 0 to sizeof(dT) do
15:           if metric.MetricType = 'SEMANTIC' then
16:             reqSemanticAnalysis(dT[seq].cT)
17:           Insert a rule for checking dT[seq].cT to metric.EvalProc
18:           mTotal = mTotal + 1
19:       Update DGM score rule for the number of mTotal metrics
20: end
```

Fig. 7 Algorithm of DGM instantiation

instances are identified. Metrics are created according to this project context. Metric calculation formulas are updated according to the number of identical metrics generated. As a result, the defined metrics are inserted into the knowledge base of the test execution engine, and the test script is generated.

Figure 7 shows the procedure of DGM instantiation. The input is ProjectContext which is the list of documents of the project and the DGM defined for the document types. The output is design guideline defined for the ProjectContext. This procedure processes every DGM defined by users. Then, it conducts the instantiation of metrics embedded in the design guideline model. The instantiation of metrics depends on the type of the metric. If the metric is 'TRACE' type, the RLIM for the metric is instantiated. If the metric is 'SEMANTIC' or 'Y/N' type, the rule of checking metric is updated for the all documents of the project. In the case of 'SEMANTIC' metrics, the content that is the target item of the metric is enlisted on the semantic analysis request queue.

Third, semantic analysis that occurs is shown in Fig. 5c. Since there is a limitation on analyzing semantics with an expert system, an external validation module verifies the generated relevance links and the contents that require semantic information. In Fig. 5c, the content with a dotted outline and all relevance links are the target of semantic analysis. After the analysis, the result shows that the content complies with its desired semantics and the RL2′ and RL3′ are not valid according to their defined semantic rules. As a result, semantic information is inserted as new rules on the test execution engine.

Finally, the engine runs a conformance test in Fig. 5d. The conformance testing shown in Fig. 5b demonstrates the visualization of the test. The relevance link information is shown as the contents with blue color, and the design guideline is shown in the red line for the relevance links and the shapes with transparent fill and red outline for the item existence and semantic analysis. The design guideline requires each of the two content instances from the documents and the relevance links RL1 and RL2 in Fig. 5. Therefore, we intuitively know that RL2′ is not satisfied and other measures are satisfied.

In actual tests, the engine executes the test script created in Fig. 5b, and all the instantiated metrics are evaluated in this stage. After the test execution, the metric score is updated on the existing test script. The engine calculates the total score of the test and the score for each design guideline. Furthermore, it imports the expert guideline from the DGM and inserts it into the test report. As a result, the engine generates an evaluation report.

4 Application

We conducted a preliminary experiment to verify the effectiveness of our method from a project we conducted previously. First, document types of interest were transformed into the RLIM. We let researchers to build the document RLIM manually for the best accuracy of the assessment at this time. Figure 8 shows the RLIM that researcher created. We set the document types of interest to the R&D plan and the SRS. Then, we identified each configuration item on both document types. In this step, each parent section has an ST–NC relationship to its child items. Finally, we let researchers make relevance links to configuration items if the researchers thought those items had relations.

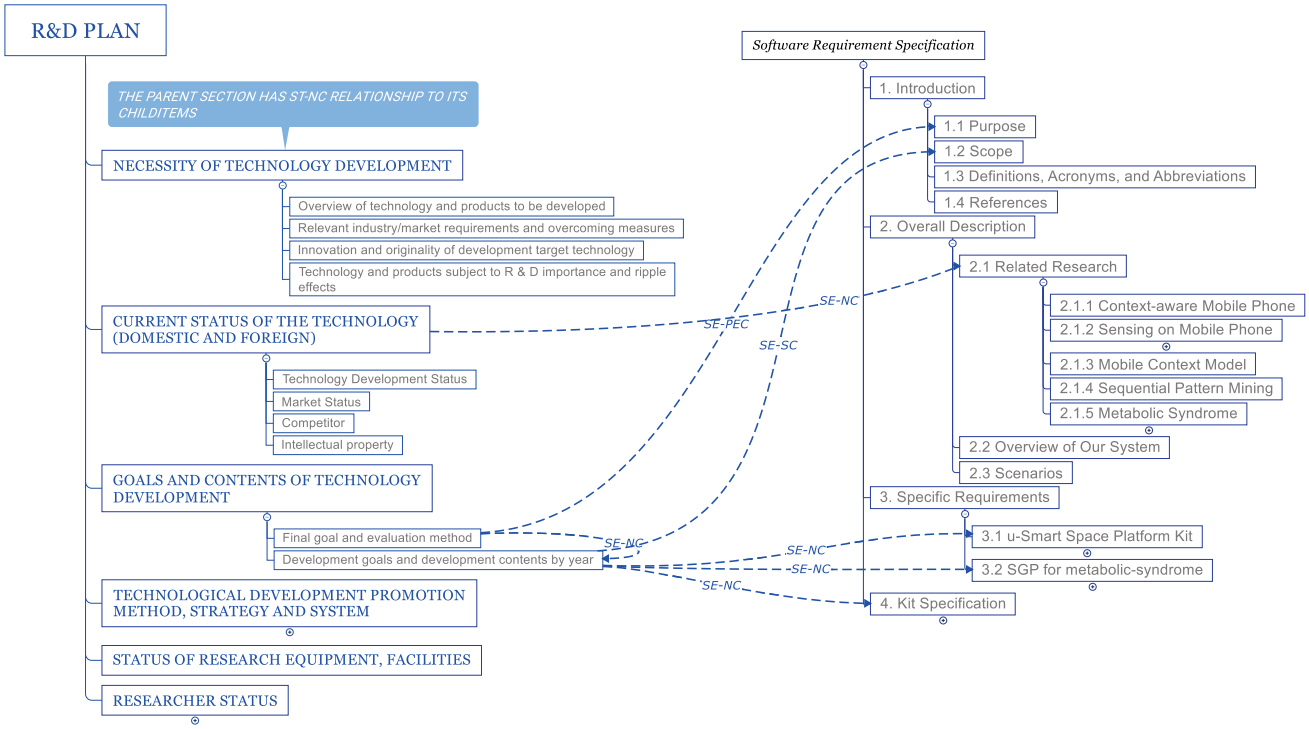

Fig. 8 An RLIM that researcher created for preliminary evaluation

Table 3 Result of preliminary application

Target document	Design guideline	Evaluation rating	Conformance level
Software	Research goal, motivation and trends	0.67	Acceptable
R&D Plan	Organization description	0.88	Good
	Proper competence for R&D process	1.00	Good
	Research strategy	0.83	Good
	Risk management	0.00	Poor
Software	Software requirement description	0.22	Poor
Requirement	Requirement analysis	0.33	Marginal
Specification	Requirement testability	0.00	poor

As regards the conformance level, *Poor* is (0.00, 0.24), *Marginal* indicates (0.24, 0.49), *Acceptable* indicates (0.50, 0.74), and *Good* indicates (0.75, 1.00)

We used the general R&D goal model for the design guideline. We selected seven solutions as the design guideline for assessment. The seven solutions have a total of 74 metrics. The solutions we selected are shown in Table 3. Then, we conducted the assessment using our proposed method. The result of the assessment displayed is shown in Table 3. Besides, the explanatory guidelines are given in Fig. 9.

Table 3 shows the assessment result of our preliminary application. Users receive an evaluation report in this form. This table shows the evaluation results of the target documents with the selected design guideline, evaluation rating, and conformance level. The software R&D plan is well written generally. However, We found problems related to the risk management process on the software R&D plan. We also found that the SRS written is of very low quality with respect to requirement specification, analysis, and testability. In this case, the ambiguity of defined requirements can be the problem when implementing the software from the requirements specification. Therefore, we need to improve the quality of the SRS for improving the R&D process.

Figure 9 shows an example of the detailed results of the evaluation results and guidelines for explanation of the evaluation. We provide users with detailed information about the

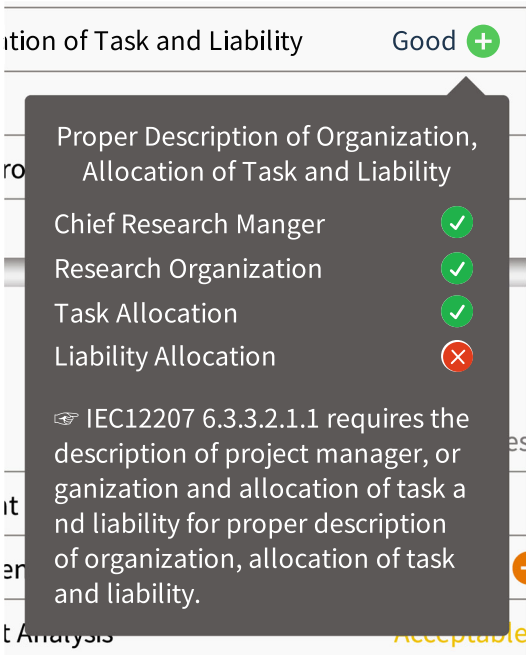

Fig. 9 Details and guidelines for evaluation results

detailed results of the evaluation and expert guidelines that explain what should be focus in documentation to provide useful feedback to users. The title shown on the top is the design guideline that measured conformance. Each item in the middle is the metric which is the minimum unit of evaluation. The comment below is the explanatory guidelines. We found that liability allocation is not properly specified in this evaluation.

5 Conclusion

In this paper, we proposed a content-based method for conformance assurance between software research documents and design guidelines. We first extracted the design guidelines from software standards. Then, we transformed the guidelines into rules for automatic evaluation. We then used an expert system to evaluate the conformance of rules obtained from the guidelines. Then, we applied our method and this shows the applicability of our method. With our method, we expect that the quality of the documents as well as the traceability between research documents will improve in a way that conforms to the best practices related to the software R&D life cycle. We expect an improvement in the document quality of research groups developing software and a reduction on the time of document quality evaluation with our proposed method. In future works, we will improve the expert guidelines so that it is more helpful to allow advice differently depending on the content of the item. we also extend the goal models for more document types such as software design specification, software test specification, test report, and technical report.

Acknowledgements This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (NRF-2014M3C4A7030504).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants performed by any of the authors.

References

Aizenbud-Reshef N, Nolan BT, Rubin J, Shaham-Gafni Y (2006) Model traceability. IBM Syst J 45(3):515–526. doi:10.1147/sj.453.0515

Allman E (2012) Managing technical debt. Commun ACM 5(5):50–55. doi:10.1145/2168796.2168798

Antonino PO, Trapp M, Barbosa P, Sousa L (2015) The parameterized safety requirements templates. In: 2015 IEEE/ACM 8th international symposium on software and system traceability, pp 29–35. doi:10.1109/SST.2015.12

Baek D, Lee B, Lee JW (2016) Content-based configuration management system for software research and development document artifacts. KSII Trans Internet Inf Syst 10(3):1404–1415. doi:10.3837/tiis.2016.03.027

Baek D, Shin JH, Lee B, Lee JW (2016b) Toward development of a traceability model measuring complicance with guidelines. In: KSII the 11th Asia Asia Pacific international conference on information science and technology, pp 37–38

Barker TT (1990) Software documentation: from instruction to integration. IEEE Trans Prof Commun 33(4):172–177. doi:10.1109/47.62811

Basili VR, Caldiera G, Rombach HD (1994) The goal question metric approach. Encycl Softw Eng 2:528–532

Bourque P, Fairley RE (2014) Guide to the software engineering—body of knowledge. doi:10.1234/12345678

Cyra L, Gorski J (2011) SCF—a framework supporting achieving and assessing conformity with standards. Comput Stand Interfaces 33(1):80–95. doi:10.1016/j.csi.2010.03.007

Dautovic A, Plosch R, Saft M (2011) Automatic checking of quality best practices in software development documents. In: 2011 11th international conference on quality software, pp 208–217. doi:10.1109/QSIC.2011.23

Fagan M, Khan MMH, Nguyen N (2015) How does this message make you feel? A study of user perspectives on software update/warning message design. Hum Centric Comput Inf Sci 5(1):36. doi:10.1186/s13673-015-0053-y

IEC (2006) IEC 62304:2006: Medical device software–software life cycle processes

ISO (2011a) ISO 26262-2:2011: road vehicles—functional safety—part 2: management of functional safety

ISO (2011b) ISO 26262-3:2011: road vehicles—functional safety—part 3: concept phase

ISO (2011c) ISO 26262-6:2011: road vehicles—functional safety—part 6: product development at the software level

ISO (2011d) ISO 26262-8:2011: road vehicles—functional safety—part 8: supporting processes

ISO/IEC (2008) ISO/IEC 12207:2008: systems and software engineering—software life cycle processes

Jain P, Verma K, Kass A, Vasquez RG (2009) Automated review of natural language requirements documents. In: Proceedings of the 2nd annual conference on India software engineering conference—ISEC '09, pp 37–45. doi:10.1145/1506216.1506224

Kelly T, Weaver R (2004) The goal structuring notation—a safety argument notation. In: Proceedings of dependable systems and networks 2004 workshop on assurance cases, Citeseer

Kim S, Lee H, Kwon H, Lee S (2015) Evaluation model of defense information systems use. J Converg 6(3):18–26

Mavin A, Wilkinson P, Harwood A, Novak M (2009) EARS (easy approach to requirements syntax). In: Proceedings of the IEEE international conference on requirements engineering, pp 317–322. doi:10.1109/RE.2009.9

Pohl K, Rupp C (2011) Requirements engineering fundamentals. Rocky Nook Inc, Santa Barbara

Sanna G, Angius A, Concas G, Manca D, Pani FE (2015) PCE: a knowledge base of semantically disambiguated contents. J Converg 6(2):10–18

Shen W, Lin CL, Marcus A (2013) Using traceability links to identifying potentially erroneous artifacts during regulatory reviews. In: 2013 7th international workshop on traceability in emerging forms of software engineering, pp 19–22. doi:10.1109/TEFSE.2013.6620149

Shin JH, Baek DS, Lee B, Lee JW (2016) Content-based conformance assurance between software research documentation and design guideline. Springer, Singapore. doi:10.1007/978-981-10-3023-9_149

Thitisathienkul P, Prompoon N (2015) Quality assessment method for software requirements specifications based on document characteristics and its structure. In: 2015 second international conference on trust system and their application, pp 51–60. doi:10.1109/TSA.2015.19

Wilson WM, Rosenberg LH, Hyatt LE (1997) Automated analysis of requirement specifications. In: Proceedings of the international conference on software engineering, pp 161–171. doi:10.1109/ICSE.1997.610237