International Journal of Advanced Computer Technology (IJACT)

ISSN:2319-7900

RECOVERING TRACE LINKS FOR SYSML MODELS USING VSM-BASED INFORMATION RETRIEVAL

Yoshihisa UDAGAWA,
Faculty of Engineering, Tokyo Polytechnic University

Abstract

Automated traceability recovery utilizing information retrieval techniques has been recognized as important for effective software development. In this paper, we discuss two approaches for augmenting the vector space model (VSM). The first approach employs document identifiers of a term, indicating where the term has been found, and a context-sensitive retrieval strategy that uses these identifiers. The other approach deals with a set of closely related terms called “family terms” in a query, which are used to adjust document vectors. We assume that requirements and design models are written in accordance with SysML specifications. Our approach can be used to check consistency among requirements and component models, including database models, screen transaction models, and process models. We perform two sets of experiments for computing the term weight using term frequency weighting and term frequency–inverse document frequency (*tf–idf*) weighting. The experimental results show that the enhanced trace retrieval models achieve higher recall and precision than the traditional VSM. In addition, our enhancements are independent of the *tf–idf* term weighting schema.

Keywords: Requirements traceability, SysML, Vector space model, Information retrieval, Family terms, Requirements engineering

1 Introduction

Maintaining traceability across software artifacts can be helpful for a number of tasks. Traceability links between requirements and a set of design documents allow developers to discover dependencies that exist among software artifacts and assess consistencies of the artifacts with respect to stated requirements.

There has been growing interest in requirements traceability in the software engineering research community, academia, and industry. The extent of traceability practice reflects a measure of software quality and process maturity. Thus, requirements traceability is mandated by many standards such as CMMI-DEV [1], systems engineering guidebooks [2] [3], and ISO/IEC/IEEE standards concerning systems and software engineering [4].

Recently, the need for automatic traceability recovery has been widely recognized. In particular, several authors have applied information retrieval techniques to automatically recover traceability links among artifacts. The rationale behind such techniques is that most software documentation is text based or contains textual descriptions. Thus, two artifacts with high textual similarity are likely reasonable tracing candidates.

Regardless of its importance, support for traceability has been inadequate. Automated traceability recovery techniques have faced major open research challenges: ways to a) extract links from relevant artifacts during software development, and b) improve the precision of retrieval results. To address these issues, we focus on:

(1) Using a standard modeling language to formalize or semi-formalize artifacts in software development.

(2) Improving accuracy of information retrieval techniques.

Obviously, the degree of similarity measures strongly depends on how accurately documents are described. Describing artifacts according to a standard is a fundamental step for successful automated traceability recovery. There are many methods to specify requirements and design artifacts. We have employed the OMG System Modeling Language (SysML) [5], which supports system development life cycles, including specification, design, verification, and validation of a broad range of systems. SysML also provides several traceability links such as refine, satisfy, and verify relationships. SysML is becoming the de facto standard in system engineering.

Maeder et al. [6] discussed a traceability link model for the requirements and implementation phases in the Unified Process. They proposed a set of rules for verification of traceability links. However this set of rules is limited to checking the pure existence of traceability links. Briand et al. [7] discussed a mechanism to model traceability between textual requirements and design models to inspect requirements–

design compliance. The approach is based on model slicing that filters out irrelevant details while maintaining sufficient context information. However, the slices may be inconclusive due to incompleteness. Delgoshaei et al. [8] proposed an approach to requirements traceability using semantic platforms in model-based systems engineering development. The proposed platform integrates traceability mechanisms with domain-specific ontologies, a set of rules for design rule checking, and artifacts modeled in SysML.

Improving information retrieval accuracy is another issue. Hayes et al. [9] used a vector space model (VSM) augmented by a thesaurus and developed a prototype tool called RETRO. Their results show that the augmented VSM outperforms a classical VSM in terms of recall, but it has the same precision performance. Several studies have discussed other information retrieval techniques, including latent semantic indexing (LSI), which is a variant of VSM, to recover traceability links among requirements, design documents, source codes, and test cases [10][11][12]. The results of these studies demonstrate the usefulness of LSI techniques for recovering traceability among software artifacts. Zisman et al. [13] presented an approach for automatic generation of traceability relations between requirements expressed in English and object models expressed in UML. Their approach results in recall and precision rates of 95% and 94% at the high end, and 46% and 52% at the low end. They used heuristic traceability rules that match syntactically related terms in the textual parts of the requirements with those of requirements object models. Cleland-Huang et al. [14] introduced three strategies to improve the performance of the dynamic requirements traceability, i.e., hierarchical modeling, logical clustering of artifacts, and semi-automatic pruning of the probabilistic network. Asuncion et al. [15] proposed an approach based on a combination of automated link capture and a machine-learning technique known as topic modeling. The learned model allows semantic categorization of artifacts and the topical visualization of the software system. Zhou et al. [16] discussed an improved VSM using requirements context information, such as title, precondition, and postcondition. They proposed a weighted knowledge model to combine the results of context analysis and functional analysis.

Although significant effort has been devoted to automatic traceability recovery, a challenging issue still remains because of the lack of accuracy. In this paper, we propose two augmentations for VSM. The first augmentation employs document identifiers of a term that indicate the term's location. The second augmentation deals with a set of closely related terms in a query, referred to as "family terms." If terms consisting of family terms are included in a document vector over a certain level, it is inferred that all terms of the family terms are included in a document vector. Our previous paper [17][18] discussed the degree of improvement by the proposed augmentations over VSM with the well-known term frequency–inverse document frequency *(tf–idf)* term weighting. In this paper, we show that our two augmentations for VSM are independent of the term weight schema, including the simple term frequency weighting and the *tf–idf* weighting. The remainder of this paper is organized as follows. In Section 2, we outline the processes for recovering traceability. In Section 3, we present SysML models for our case study. In Section 4, we discuss augmented information retrieval models, and in Section 5, we present the results of our experiments. In Section 6, we evaluate the results using recall and precision. Finally, Section 7 concludes the paper.

2 Overview

2.1 Process for Recovering Traceability

Figure 1 shows the major process flow for recovering traceability discussed in this paper. Our study deals with traceability among artifacts in requirements and those in design phases. The process flow consists of SysML diagrams and two tools, i.e., a document analyzer and a traceability generator. Artifacts are divided into two categories: requirements artifacts and design artifacts. For illustrative purposes, Figure 1 contains only the diagrams (requirements, data models, state machine diagrams, and activity diagrams) necessary to describe the example in this paper. However, any SysML/UML diagram can be added as required.

First, design experts create design artifacts that satisfy a set of requirements. The experts also declare true links among the requirements and the design artifacts using the SysML refine relationship (solid lines in Figure 1). Second, information retrieval is applied to the requirements and the design artifacts to generate the recovered links (dashed lines in Figure 1). Third, the true links and the recovered links are compared to check the extent to which the true links are realized in the design artifacts. The results of the comparison facilitate evaluation of the quality of artifacts in terms of traceability.

2.2 Document Analyzer

The document analyzer extracts three categories of terms for use in the traceability generator: key terms, extracted main terms, and family terms.

(1) Key terms are unique identifiers for recognizing a primitive element in the requirements and design diagrams. In this paper, the key terms are preceded by a sharp symbol "#." We assume that key terms are used in a consistent manner throughout the software development life cycle and contribute to an element of a document vector.

The key term can consist of terms connected by an underscore "_." In this case, the term is decomposed into each of its constituents that are treated as extracted main terms. Thus, a key term generally has twice the weight of an extracted main term.

(2) The extracted main terms include verbs, nouns, adjectives, and adverbs. Prepositions and conjunctions are excluded. An extracted main term contributes to an element of a document vector. Note that the document analyzer relates each key term and extracted main term to documents from which the term has been extracted. For example, if a term is extracted from the requirements diagram, then the term is tagged with the document identifier "requirement."

(3) A family term is a set of the extracted main terms in a sentence consisting of a query. For example, if a sentence consists of the extracted main terms A, B, C, and D, then a set of the terms forms a family term. If a query consists of more than one sentence, then more than one family term can be generated.

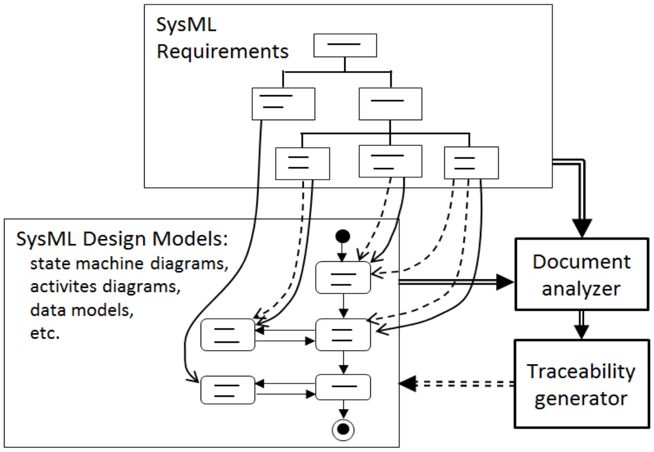


Figure 1. Process flow for recovering traceability

Solid lines: True links declared by designers
Solid double lines: Textual data exitacted from SysML models
Dashed lines: Links recoverd by information retrieval
Double dashed lines: A set of recoverd links

2.3 Traceability Generator

Using the terms extracted by the document analyzer, the traceability generator computes the similarities between a query and the documents generated from SysML diagrams. The generator is based on VSM [19] and features three modes: (1) a simple vector space mode, (2) a mode using the terms attributed by the document identifier (attributed term mode), and (3) a mode using the family terms.

In the attributed term mode, terms are dropped according to the context of traceability recovery based on the document identifiers. For example, when a user intends to recover traceability from requirements to design artifacts, we can drop those terms that have only the document identifier "requirement" from the query vector.

Document vectors concerning trace recovery are rewritten on the basis of the family terms. For example, if the extracted main terms {A, B, C, D} are found in a sentence of the query, then they are treated as family terms. The family terms are successively applied to documents. If a document contains {A, B, C}, then it is inferred that it is possible that term D is contained in the document. During the initial study, the possibility of an inferred term is set to 1, meaning that the term is definitely included. In addition, we assume that a set of family terms is activated when more than half of the terms are included in a document vector.

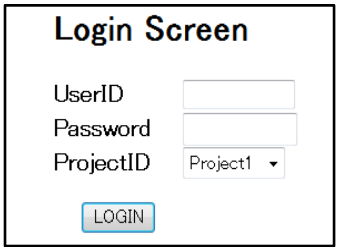
3 Modeling in SysML

3.1 Login Function for a Web System

A login function for a business application is vital for identifying the user of the application and protecting the application from unauthorized access. The concept of a login function is fairly simple. It logs in a user after verifying the user ID and password by referring to a table in a database that contains valid pairs of user IDs and passwords. However, in practice, a designer must consider support functions, such as allowing the user to change passwords, managing expiration of passwords, and setting rules regarding the syntax of the password.

In a typical web application that manages project artifacts, users consist of two classes: managers and designers. A user can participate in one or more projects. The user might be a manager of some projects and a designer of other projects. The role of the user in a project is stored in a database, which is described in detail in Subsection 3.3.

The layout of the login screen is shown in Figure 2. The user who wants to login enters a user ID and a password in the corresponding fields, and chooses a project name from the drop-down list.



Login Screen

UserID

Password

ProjectID Project1

Figure 2. Layout of the login screen

3.2 Requirements of the Login Function

As mentioned in the previous subsection, the requirements of the login function should be specified in the context of the operation and maintenance of user IDs and passwords. The requirements of this case study are summarized as follows.

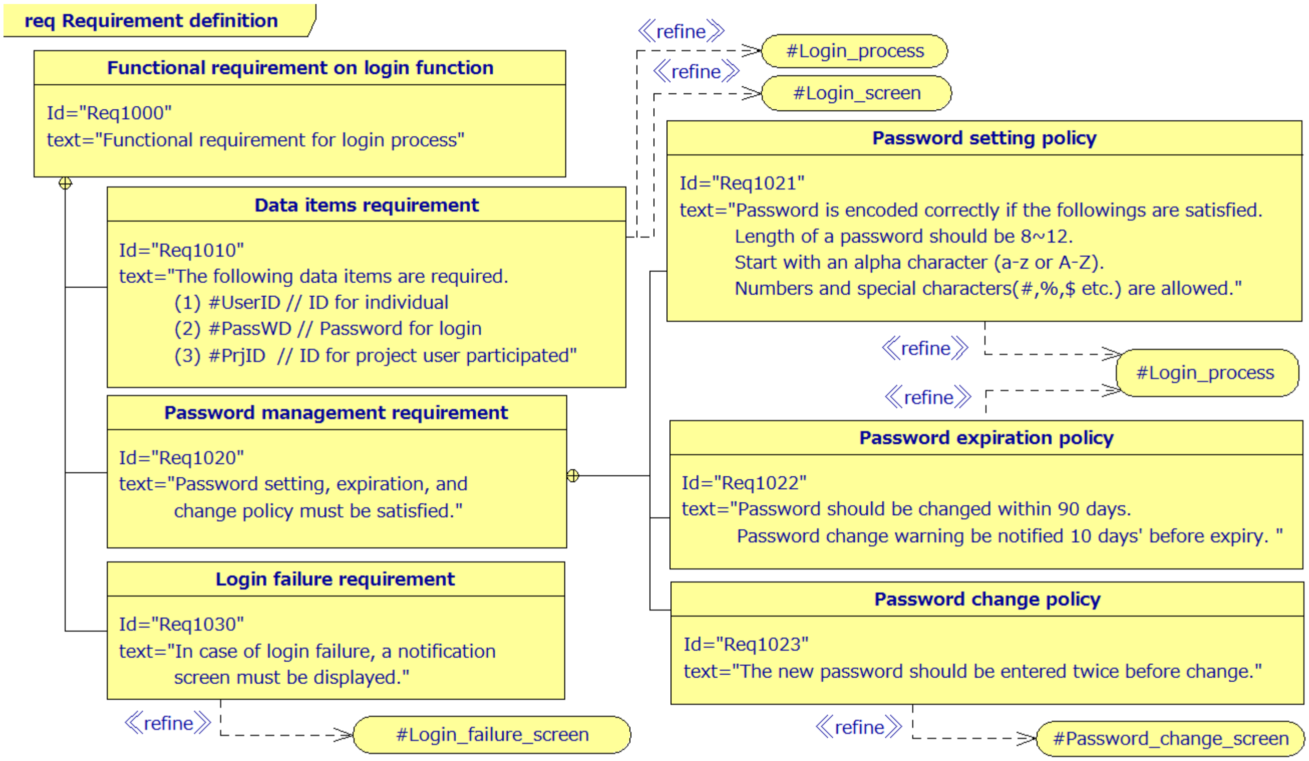
(1) Data items requirement: User ID, password, and project ID should be entered.

(2) Password management requirement: For security purposes, a password should be robust in its combination of characters, and the user should periodically change his or her password. This requirement is further decomposed into a password setting policy, a password expiration policy, and a password change policy.

(3) Login failure requirement: In the case of login failure, a notification must be displayed.

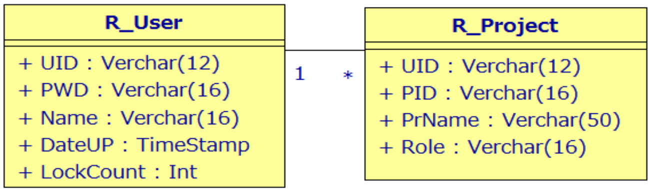
The SysML requirements diagram for these requirements is shown in Figure 3. Each requirement is attached by the <<refine>> relationship to the other corresponding SysML design models, which are described in the following subsections. Note that <<refine>> relationships are defined manually as true links by the experts who designed the login function. All <<refine>> relationships are successfully retrieved by our augmented retrieval models. This is discussed in further detail in Section 5.

The aim of the database design is to define data items, relationships, and data constraints for the application. The entities "user" and "project" are identified. The entity "user" has the attributes user ID, password, user name, and the date of the last password update. The entity "project" has the attributes project ID, project name, and project members and their roles. The entities "user" and "project" are translated into the tables R_user and R_project, respectively. A user can participate in one or more projects; therefore, there is a one-to-many relationship between the "user" and "project" entities. The relationship between these entities is specified by the user ID as a foreign key in the table R_project, as is shown in Figure 4, which is depicted according to a UML class profile for data modeling [20]. Note that table names and data item names are processed as key terms without the preceding a # symbol.


req Requirement definition
Functional requirement on login function
Id="Req1000"
text="Functional requirement for login process"
Data items requirement
Id="Req1010"
text="The following data items are required.
(1) #UserID // ID for individual
(2) #PassWD // Password for login
(3) #PrjID // ID for project user participated"
Password management requirement
Id="Req1020"
text="Password setting, expiration, and
change policy must be satisfied."
Login failure requirement
Id="Req1030"
text="In case of login failure, a notification
screen must be displayed."
《refine》
#Login_failure_screen
《refine》
#Login_process
《refine》
#Login_screen
Password setting policy
Id="Req1021"
text="Password is encoded correctly if the followings are satisfied.
Length of a password should be 8~12.
Start with an alpha character (a-z or A-Z).
Numbers and special characters(#,%,$ etc.) are allowed."
《refine》
#Login_process
《refine》
Password expiration policy
Id="Req1022"
text="Password should be changed within 90 days.
Password change warning be notified 10 days' before expiry. "
Password change policy
Id="Req1023"
text="The new password should be entered twice before change."
《refine》
#Password_change_screen

3.3 Database Model

ISSN:2319-7900

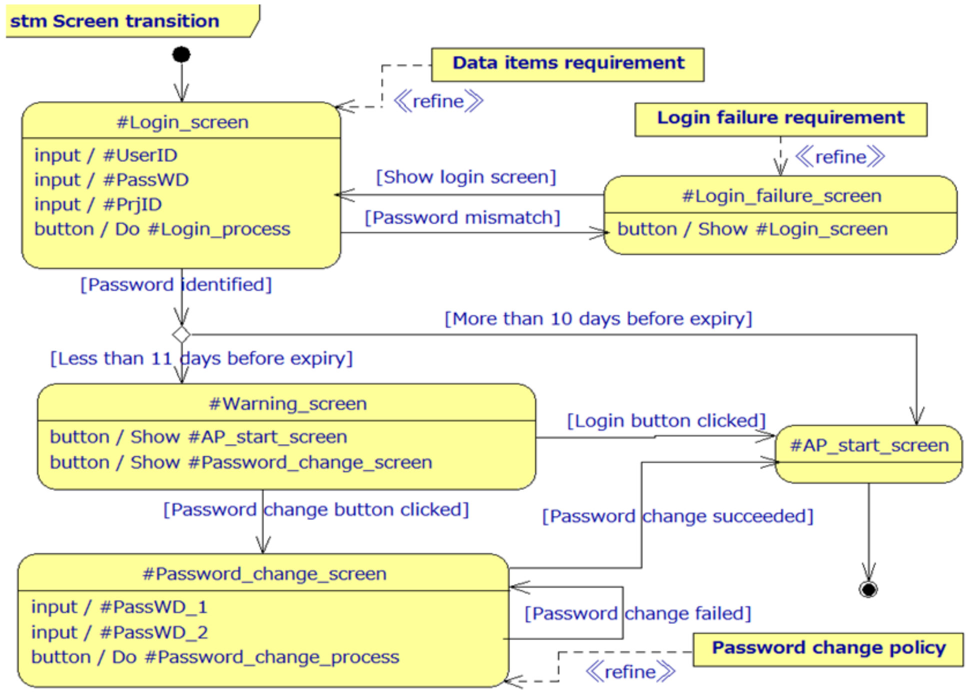


3.4 Screen Transition Model

The state machine diagram represents behavior as the state history of an object in terms of its states and transitions. Screen transitions can be represented by a SysML state machine diagram, in which the screens correspond to the states. By analyzing the requirements of the login function, five screens are recognized, i.e., #Login_screen, #Login_failure_screen, #Warning_screen, #Password_change_screen, and #AP_start_screen. Figure 5 shows the overall screen transitions and depicts the screens, events, and the <<refine>> relationships that are manually defined as true links.

3.5 Process Model

The SysML activity diagram is a graph-based diagram showing flow of control, and is therefore used to define processes that are associated with screen transitions. Figure 6



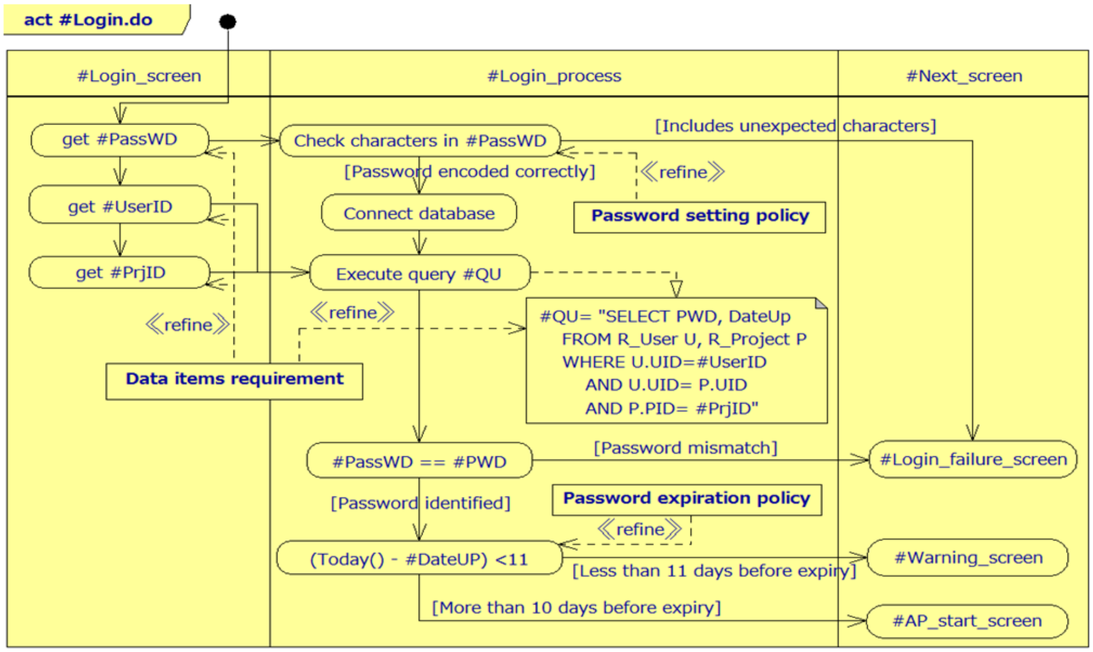


Figure 6. Process diagram

ISSN:2319-7900

shows the login process launched when the button on the login screen is clicked. The main part of the process is described in the central lane, while the login and next screens are described in the left and right lanes, respectively. Note that the <<refine>> relationships represent true links.

4 Rcovering Requirements Traceability

4.1 Defining Documents

The SysML diagrams in Figures 3, 5, and 6 are decomposed into documents forming the basic elements of trace information retrieval. In our case study, we define the documents as follows:

(1) All leaf nodes of the requirements hierarchy in Figure 3 are decomposed into five documents, i.e., "Req1010," "Req1021," "Req1022," "Req1023," and "Req1030."

(2) All screen nodes of the screen transition diagram in Figure 5 are decomposed into five documents.

(3) The three lanes in Figure 6, i.e., #Login_screen, #Login_process, and #Next_screen, are decomposed into three documents.

4.2 Analyzing SysML Diagrams

The document analyzer parses all documents in preparation for information retrieval based on VSM. The documents are indexed using the terms extracted from the documents themselves. The extraction of the terms and the indexing of the documents are performed according to the following steps.

(1) All documents are tokenized. Furthermore, all capital letters are transformed into lower case letters.

(2) The terms preceded by a # symbol are processed as key terms. In addition, the terms consisting of key terms connected by underscores are divided into each of the terms that are treated as either stop words or extracted main terms.

(3) Stop words (i.e., words that are not useful for the purpose of retrieval, such as articles, pronouns, prepositions, conjunctions, and numbers) are removed.

(4) The stems of the remaining words are extracted to ensure that different forms of the same term are treated as one, i.e., converting plurals into singulars and transforming conjugated verb forms into infinitives.

(5) Each term has document identifiers that indicate the documents from which the term has been extracted. The identifier is further categorized as "requirement" or "design." When a term occurs in both requirements documents and design documents, it can be attributed as {"requirement," "design"}. The document identifiers are used to augment information retrieval, which is discussed in Subsection 4.4. A requirement can contain sub-requirements, thus forming a requirements hierarchy. The title of each requirement element contributes to form a document vector in an accumulated manner along the requirements hierarchy.

4.3 Vector Space Model

The vector space model (VSM) [19] is an algebraic model for representing text documents as vectors of identifiers or terms. Given a set of documents *D*, a document d_j in *D* is represented as the vector of term weights:

$$d_j = \left(w_{1,j}, w_{2,j}, \ldots, w_{N,j}\right), \quad (1)$$

where N is the total number of unique terms in the document set *D*, and $w_{i,j}$ is the weight of the *i*-th term according to a certain weighting scheme.

A user query is also converted into a vector *q* that is similar to the document vector.

$$q = \left(w_{1,q}, w_{2,q}, \ldots, w_{N,q}\right) \quad (2)$$

The similarity between the document d_j and the query *q* can be computed as the cosine of the angle between the two vectors d_j and *q* in N-dimensional space as follows:

$$\text{Similarity}\left(d_j, q\right) =$$

$$\cos\left(d_j, q\right) = \frac{\sum_{i=1}^{N} w_{i,j} * w_{i,q}}{\sqrt{\sum_{i=1}^{N} w_{i,j}^2} * \sqrt{\sum_{i=1}^{N} w_{i,q}^2}}, \quad (3)$$

Several different methods for computing term weights have been developed. Term frequency, which is the simplest way to define a term weight, is usually defined on the basis of the number of term occurrences in a document. Another is the well-known *tf–idf* method. The weight $w_{i,j}$ is computed as follows:

$$w_{i,j} = tf_{i,j} \cdot idf_i, \quad (4)$$

where $tf_{i,j}$ is the *i*-th term frequency in a document d_j and is usually the number of term occurrences in the document. idf_i is the inverse document frequency of the *i*-th term in the document set *D*. Inverse document frequency is computed as follows:

$$idf_i = log_2\left[\frac{m}{df_i}\right], \quad (5)$$

where df_i is the number of documents in which the *i*-th term occurs and *m* is the total number of documents. idf_i is a weight for the frequency of the term in a set of documents. Equation (5) dictates that terms with fewer occurrences in a set of documents are treated as more informative.

4.4 Augmenting the Vector Space Model

We augmented VSM based on a similarity computation using the document identifiers and a set of closely related terms as follows.

(1) The terms are dropped according to the context of traceability recovery based on the document identifiers. For

example, when a user intends to recover traceability from requirements to design artifacts, we can drop terms in the query that only have the document identifier "requirement" from the query vector. In theory, terms found only in the requirements do not match those only used in the designs and vice versa. As described in Section 5, the "data items requirement" in Figure 3 is used as a query. In this case, the weights of the terms in the query, including "function," "item," "individual," "participate," "project," and "require" are set to 0 because these terms are attached by the document identifier "requirement." In other words, they are not used in any of the document vectors generated from Figures 5 and 6. Adjustment of the query vector increases the similarity values. The effects of the document identifiers are described in Section 5.

(2)Closely related terms, i.e., family terms, are treated as a set. For example, if the extracted main terms {A, B, C, D} are found in a query, then they are treated as family terms. The family terms are successively applied to documents. If a document contains {A, B, C}, then it is inferred that it is possible that the term D is also contained in the document. During the initial study, the possibility of an inferred term is set to 1, meaning that the term is definitely included. In addition, we assume that a set of family terms is activated when more than half of the terms are included in a document vector. As described in Section 5, "data items requirement" in Figure 3 is used as a query. In this case, a set of terms, including {"#PassWD," "#PrjID," "#UserID," "data," "id," "login," "password," "user"}, is extracted as family terms. The family terms are successively applied to documents. The document vectors corresponding to #Login_screens in Figures 5 and 6 are matched, and the values for "data" and "password" are updated to 1. Updating the document vectors and adjustment of the query vector using the document identifier increase the similarity values. The effects of the family terms are described in Section 5.

5 Result of the Information Retrieval

5.1 Overview of Experiments

We conducted experiments on documents translated from the three diagrams shown in Figures 3, 5, and 6. A total of 82 main terms, including 16 key terms attributed by the document identifiers, were extracted from the diagrams.

First, we conducted two sets of experiments for tracing requirements to designs by computing the term weight using term frequency weighting and *tf–idf* weighting. The experimental details are described in Subsection 5.2. Next, we conducted two other sets of experiments for tracing designs to requirements by computing the term weight using term frequency weighting and *tf–idf* weighting. The experimental details are described in Subsection 5.3. The abbreviations VS, VA, and VF+VA denote the simple vector space mode, the mode using the attributed terms, and the mode using both family terms and attributed terms, respectively.

5.2 Tracing Requirements to Designs

5.2.1 Experiments using Term Frequency Weighting

Each of the five documents translated from the requirement process shown in Figure 3 was used as a query vector. We performed five types of retrievals to trace requirements to designs by computing the term weight using term frequency weighting in the three modes, i.e., VS, VA, and VF+VA, resulting in 15 retrievals.

Figure 7 shows the retrieved documents with the similarity values obtained with term frequency weighting. For example, the similarity values corresponding to #Login_screen (F) in Figure 5 rise from 0.355 to 0.495 using the document identifiers (VA), and rise from 0.355 to 0.676 using the family terms and the document identifiers (VF+VA).

In Figure 7, the circled documents indicate the manually defined true links. Note that the similarity value of the VF+VA mode is not less than the values obtained from the other modes in all the retrievals. In several cases, the differences between the similarity values of the true links and the non links increased.

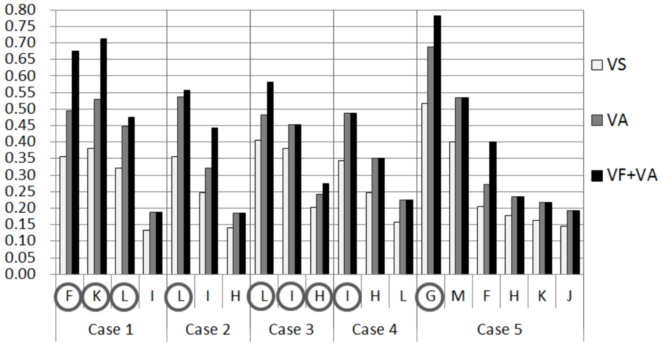


Figure 7. Similarity values of the retrieved documents: tracing requirements to designs using term frequency weighting

F: #Login_screen	**J: #AP_start_screen**
G: #Login_failure_screen	**K: #Login_screen**
H: #Warning_screen	**L: #Login_process**
I: #Password_change_screen	**M: #Next_screen**

The similarity value of a document reflects how precisely the document is described using the terms in a query. In this case study, the similarity value of the warning screen, i.e., document H of case 3, is the smallest true link and is less than half the value of the largest one (G of case 5). This indicates that descriptions concerning the warning screen require revision. This is a typical example that shows how requirements tracing contributes to estimating the quality of documents.

5.2.2 Experiments using *tf–idf* Weighting

We performed five types of retrievals to trace requirements to designs using *tf–idf* weighting in the three modes. Figure 8 depicts the retrieved documents with the similarity values. For example, the similarity values corresponding to #Login_screen (F) in Figure 5 rise from 0.286 to 0.398 using the document identifiers (VA), and rise from 0.286 to 0.540 using the family terms and the document identifiers (VF+VA).

In all cases, the similarity values decrease compared to the values in Figure 7 because of the *tf–idf* weighting. However, the order of the retrieved documents ranked by the similarity values remains unchanged with the exception documents J and K in case 5.

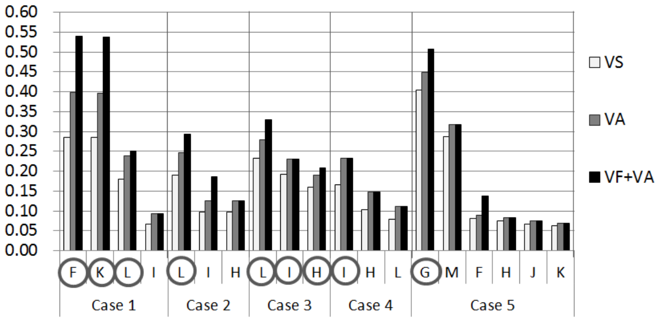


Figure 8. Similarity values of the retrieved documents: tracing requirements using designs in *tf–idf* weighting

F: #Login_screen **J: #AP_start_screen**
G: #Login_failure_screen **K: #Login_screen**
H: #Warning_screen **L: #Login_process**
I: #Password_change_screen **M: #Next_screen**

5.3 Tracing Designs to Requirements

5.3.1 Experiments using Term Frequency Weighting

Each of the eight documents translated from the design diagrams in Figures 5 and 6 was used as query vectors. We conducted eight types of retrievals to trace designs to requirements using the term frequency weighting in the three modes, thus resulting in 24 retrievals.

Figure 9 shows the retrieved documents with the similarity values obtained using term frequency weighting. The circled documents indicate the true links. Document E of case 5 and document E of case 8 denote non links. The former is apparently a non link because its similarity value is approximately half as large as the value of the average true links. However, it is difficult to determine whether the latter document is actually a non link. "Login failure requirement" in Figure 3 and #Next_screen in Figure 6 share three extracted main terms: "login," "failure," and "screen." These terms are generally used, and they characterize neither the document nor the query. This non link is primarily due to the way that the process diagram in Figure 6 and the contents of the document description are decomposed.

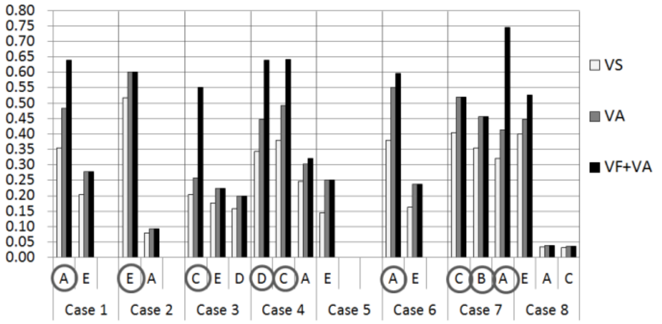


Figure 9. Similarity values of the retrieved documents: tracing designs to requirements using term frequency weighting

A: Data items requirement **D: Password change policy**
B: Password setting policy **E: Login failure requirement**
C: Password expiration policy

5.3.2 Experiments using *tf–idf* Weighting

We also examined the ability to trace designs to requirements by computing the term weight using *tf–idf* weighting. Figure 10 shows the retrieved documents with the similarity values. The results are similar to those of Figure 9, with the exception of documents B and E in case 3 and documents A and B in case 8, which represent non links.

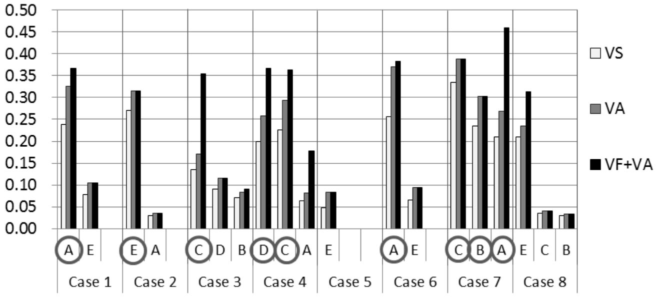


Figure 10. Similarity values of the retrieved documents: tracing designs to requirements using *tf–idf* weighting

A: Data items requirement **D: Password change policy**
B: Password setting policy **E: Login failure requirement**
C: Password expiration policy

6 Evaluation

6.1 Recall and Precision

ISSN:2319-7900

The results were evaluated using two widely used metrics: recall and precision. Recall is the ratio of the number of correct documents retrieved for a given query to the number of correct documents for that query. Recall can be expressed as follows:

$$\text{Reall} = \frac{|\,\boldsymbol{Corr_{doc}} \cap \boldsymbol{Ret_{doc}}\,|}{|\,\boldsymbol{Corr_{doc}}\,|}, \qquad (6)$$

where $Corr_{doc}$ and Ret_{doc} represent the correct documents and retrieved documents using an information retrieval technique, respectively. In this study, the number of correct documents is obtained from the true links that are defined by design experts using the SysML <<refine>> relationships.

Precision is the ratio of the number of correct documents retrieved to the number of documents retrieved. The number of documents retrieved is obtained from the result of information retrieval. Precision can be expressed as follows:

$$\text{Precision} = \frac{|\,\boldsymbol{Corr_{doc}} \cap \boldsymbol{Ret_{doc}}\,|}{|\,\boldsymbol{Ret_{doc}}\,|}. \qquad (7)$$

6.2 Tracing Requirements to Designs

Figure 11 shows recall and precision for retrieving the design documents regarding the requirements identified by computing the term weight using term frequency weighting. Figure 11 is obtained from Figure 7 by examining the similarity value in increments of 0.025. The vertical axis represents recall and precision, and the horizontal axis represents the similarity value. Recall declines sharply when the similarity value is greater than 0.35 in the VS mode. The recall stays at approximately 90% up to a similarity value of 0.45 in the VA and VF+VA modes. Precision is greater than 60% for the three modes when the similarity value is greater than 0.20.

Here, we analyze in detail the 67% line of recall and precision. In the VS mode, the recall reaches values greater than 67% at a similarity value of 0.188, whereas the precision is maintained at greater than 67% when the similarity value is lesser than 0.35. Trace retrieval achieves values greater than 67% for both recall and precision in the similarity value range of 0.188–0.35. In summary, the following shows the ranges of similarity values for each of the three VSMs in which both recall and precision were maintained at greater than 67%.

VS mode: 0.188–0.350 (0.162 width)
VA mode: 0.238–0.475 (0.237 width)
VF+VA mode: 0.238–0.475 (0.237 width)

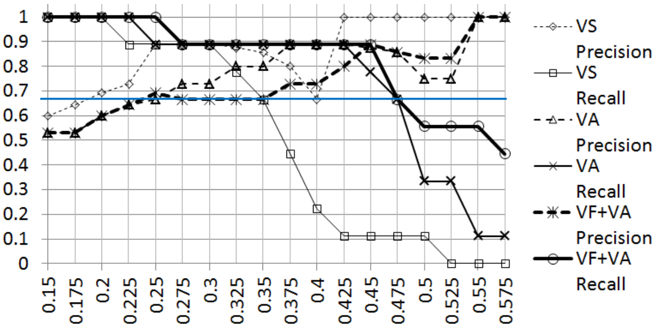


Figure 11. Recall and precision for tracing requirements to designs using term frequency weighting

Figure 12 shows recall and precision for retrieving design documents regarding the requirements identified by computing the term weight using *tf–idf* weighting. Figure 12 is obtained from Figure 8 by examining the similarity value at increments of 0.025. Recall declines sharply when the similarity value is greater than 0.175 in the VS mode and is greater than 0.225 in the VA and VF+VA modes. Precision becomes greater than 60% for all three modes when the similarity value is greater than 0.10.

The following shows the ranges of similarity values in the three modes where both recall and precision are maintained at greater than 67%.

VS mode: 0.110–0.185 (0.075 width)
VA mode: 0.115–0.240 (0.125 width)
VF+VA mode: 0.120–0.238 (0.118 width)

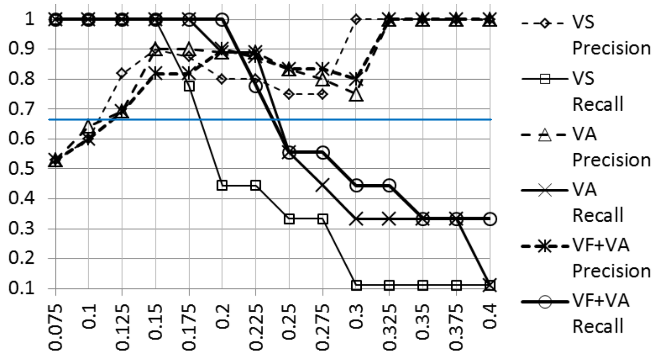


Figure 12. Recall and precision for tracing requirements to designs using *tf–idf* weighting

6.3 Designs to Requirements

Figure 13, obtained from Figure 9, shows the recall and precision for retrieving the requirements concerning the design documents using term frequency weighting. Figure 14, obtained from Figure 10, shows the recall and precision for retrieving the requirements using *tf–idf* weighting. Figures 13 and 14 show that recall declines first in the VA mode and last in the VF+VA mode.

The following shows the ranges of the similarity values in Figure 13 for each of the three VSMs in which both recall and precision were maintained at greater than 67%.
VS mode: 0.167–0.350 (0.183 width)
VA mode: 0.234–0.450 (0.216 width)
VF+VA mode: 0. 234–0.575 (0.341 width)

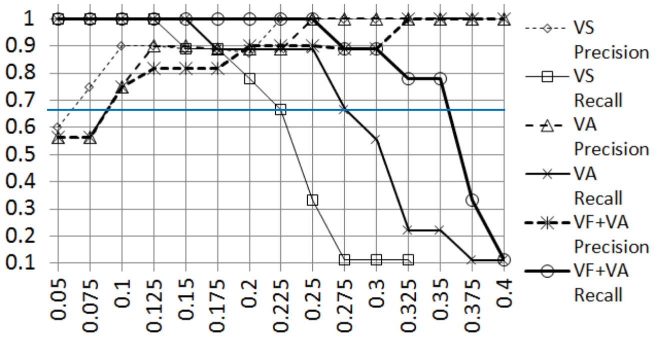


Figure 13. Recall and precision for tracing designs to requirements using term frequency weighting

The following shows the ranges of the similarity values in Figure 14 in the three modes where both recall and precision are maintained at greater than 67%.
VS mode: 0.060–0. 225 (0.165 width)
VA mode: 0.087–0. 275 (0.188 width)
VF+VA mode: 0.087–0. 355 (0.268 width)

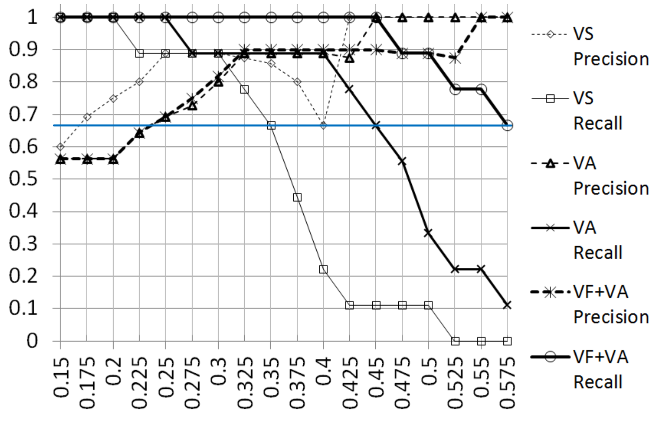


Figure 14. Recall and precision for tracing designs to requirements using *tf–idf* weighting

6.4 Summary of Evaluation

Figure 15 summarizes the range widths of the similarity values of bidirectional retrieval experiments in all three modes using the two term weighting methods. The vertical axis represents the ratio of the range widths of the three modes to those of the VS mode. Figure 15 shows that the ranges of the similarity values expanded by the augmentations in all four sets of experiments, and the range ratios are independent of the term weighting schema. Note that a wider range of similarity values indicates that retrieval results are less affected by the threshold of the similarity values, thus yielding more stable retrieval results. In this case study, the results of the similarity value ranges indicate that the VF+VA mode outperforms the other modes.

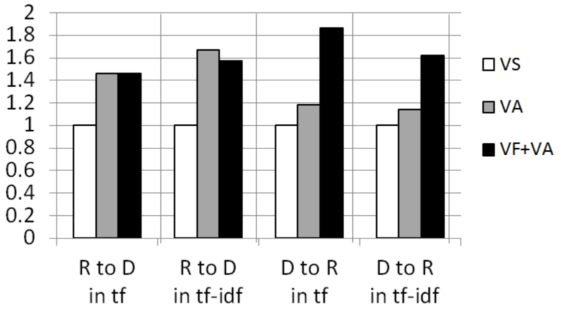


Figure 15. Summary of the range ratios of the similarity values to the VS mode. "tf" denotes term frequency weighting and "tf-idf" denotes *tf–idf* weighting.

7 Conclusions and Future Work

The primary aim of our proposed information retrieval technique is to introduce context information to enhance the accuracy of candidate trace link lists. The first enhancement uses a term with the document identifier indicating the document from which the term has been extracted. The other enhancement utilizes a set of closely related terms called "family terms" that are extracted from a query.

We employed SysML as a tool for documenting because it is a general-purpose modeling language and has been applied to various projects in system engineering. SysML provides several requirements relationships. We used the refining requirements relationship to declare "true links," which are set by design experts. The results of our information retrieval technique were evaluated against the "true links." The results of our experiments show that the proposed approach outperforms the traditional vector space model approach in recall and precision and that our enhancements are independent of the *tf–idf* term weighting schema.

In this paper, we used only four types of diagrams, i.e., requirements, data modeling, state machine, and activity diagrams. However, our information retrieval technique is independent of the documentation tool; thus, any SysML/ UML diagram can be added as required.

In future, we plan to broaden the applicability of the information retrieval models to trace links between SysML models and source codes. These are the major artifacts of system development; therefore, automated tracing between them will allow software developers to better understand systems.

ISSN:2319-7900

Acknowledgments

We would like to thank Hisayuki Masui, Yohtaro Miyanishi, and Tamotsu Noji for their suggestions on software system modeling.

References

[1] CMMI Product Team, "CMMI for Development, Ver. 1.3 - Improving processes for developing better products and services," CMU/SEI-2006-TR-008, Software Engineering Institute, Carnegie Mellon University, November 2010. http://www.sei.cmu.edu/.

[2] California Department of Transportation, "Systems Engineering Guidebook for ITS, Ver. 3.0," November 2009. http://www.fhwa.dot.gov/cadiv/segb/.

[3] INCOSE, "Systems Engineering Handbook, Ver. 3 - A Guide for System Life Cycle Processes and Activities," pp. 1-185, June 2006.

[4] ISO/IEC/IEEE, "Std 12207-2008 - Standard for Systems and Software Engineering - Software Life Cycle Processes," pp. 1-138, January 2008.

[5] Object Management Group, "OMG Systems Modeling Language, Ver. 1.3," June 2012. http://www.omgsysml.org/.

[6] P. Maeder, I. Philippow, and M. Riebisch, "A Traceability Link Model for the Unified Process," The 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pp. 700-705, July 2007.

[7] L. Briand, D. Falessi, S. Nejati, M. Sabetzadeh, and T. Yue, "Traceability and SysML Design Slices to Support Safety Inspections: A Controlled Experiment," Journal ACM Transactions on Software Engineering and Methodology (TOSEM), Vol. 23, No. 1, Article No. 9, February 2014.

[8] P. Delgoshaei, M. A. Austin, and D. A. Veronica, "A Semantic Platform Infrastructure for Requirements Traceability and System Assessment," The 9th International Conference on Systems (ICONS 2014), pp. 215-219, February 2014.

[9] J. H. Hayes, A. Dekhtyar, S. K. Sundaram, and S. Howard, "Helping Analysts Trace Requirements: An Objective Look," The 12th International Requirements Engineering Conference, pp. 249-259, September 2004.

[10] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, "Recovering Traceability Links between Code and Documentation," IEEE Transactions on Software Engineering, Vol. 28, No. 10, pp. 970-983, October 2002.

[11] A. Marcus and J. I. Maletic, "Recovering Documentation-to-Source-Code Traceability Links Using Latent Semantic Indexing," The 25th International Conference on Software Engineering, pp. 125-135, May 2003.

[12] M. Lormans and A. van Deursen, "Can LSI help Reconstructing Requirements Traceability in Design and Test?," The 10th European Conference on Software Maintenance and Reengineering, pp. 47-56, March 2006.

[13] A. Zisman, G. Spanoudakis, E. Perez-Minana, and P. Krause, "Tracing Software Requirements Artifacts," The International Conference on Software Engineering Research and Practice, pp. 448-455, June 2003.

[14] J. Cleland-Huang, R. Settimi, C. Duan, and X. Zou, "Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability," The 13th International Requirements Engineering Conference, pp. 135-144, August 2005.

[15] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor, "Software Traceability with Topic Modeling," The 32nd ACM/IEEE International Conference on Software Engineering, pp. 95-104, May 2010.

[16] J. Zhou, Y. Lu, and K. Lundqvist, "A Context-based Information Retrieval Technique for Recovering Use-Case-to-Source-Code Trace Links in Embedded Software Systems," The 39th Euromicro Conference on Software Engineering and Advanced Applications, pp. 252-259, September 2013.

[17] Y. Udagawa, "An Augmented Vector Space Information Retrieval for Recovering Requirement Traceability," The 11th IEEE International Conference on Data Mining Workshops, pp. 771-778, December 2011.

[18] Y. Udagawa, "Enhancing Information Retrieval to Automatically Trace Requirement and Design Artifacts," The 13th ACM International Conference on Information Integration and Web-based Applications & Services, pp. 292-295, December 2011.

[19] G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," Information Processing and Management, Vol. 24, No. 5, pp. 513-523, 1988.

[20] W. A. Scott, "A UML Profile for Data Modeling," 2009. http://www.agiledata.org/essays/umlDataModelingProfile.html.

Biographies

The author received a Ph.D. in aeronautical science from the University of Tokyo in 1982. From 1982 to 2010, he worked on various project related to database and Web-based system development for industrial use. Since April 2010, he has served as a professor in the computer science department at Tokyo Polytechnic University. His research interests include data mining, case-based reasoning, and quality management of systems development. Prof. Udagawa may be reached at udagawa@cs.t-kougei.ac.jp