TRANSFORMATION OF ORGANIZATION OF SOFTWARE REQUIREMENTS SPECIFICATIONS

Yusuke Matsuo and Atsushi Ohnishi
Department of Computer Science, Ritsumeikan University, 1-1-1 Noji-Higashi, Kusatsu, Shiga 525-8577, Japan

Keywords: Organization of Software Requirements Specification, Transformation of Software Requirements Specification.

Abstract: Specific software functional requirements can be organized with different ways, such as user class oriented organization, functional hierarchy oriented organization, stimulus oriented organization, and so on. User class oriented software requirements specification is easy to understand for each class of the user's behaviours, but difficult to understand functional hierarchy. We adopt a controlled requirements language named X-JRDL as a requirements language and propose a transformation method between two software requirements specifications organized in different ways.

1 INTRODUCTION

In software development and maintenance, a software requirements specification (SRS) is often referred for revising requirements, designing software, and modifying the specification. Specific requirements of an SRS can be organized with different ways (IEEE 1998). In (IEEE 1998) eight templates of specific requirements of an SRS are proposed. These are two temples organized by mode, a template organized by user class, a template organized by object, a template organized by feature, a template organized by stimulus, a template organized by functional hierarchy, and a template by multiple organizations. Even if each of differently organized SRSs specifies exactly the same software requirements, the understandability of them are different each other.

For example, user oriented SRSs may be organized by user class, so readers who want to know what kind of functions should be used by a certain user and what kind of inputs should be entered by a certain user prefer SRSs organized by user class, while the readers cannot easily know them in case of SRSs organized by functional hierarchy. If an SRS organized by functional hierarchy can be transformed into an SRS organized by user class, the above problem can be solved, but manual transformation of SRSs with different ways is difficult in case of large SRSs especially.

Actually, only the section 3.2 of these templates are different, but other sections are same each other (IEEE 1998). The authors propose a method to automatically transform SRSs of different organizations and have developed a prototype system based on the method.

2 REQUIREMEMTS LANGUAGE: X-JRDL

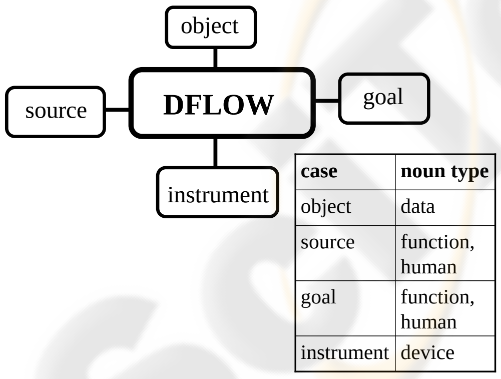
We developed requirements model named Requirements Frame and a text-base requirements language named X-JRDL based on the model (Ohnishi 1996). In this research we adopt X-JRDL as a requirements language, since it is quite easy to transform SRSs with X-JRDL organized differently.

Since X-JRDL aims to specify requirements of file-oriented applications, this language provides 6 noun types (human, function, file, data, control, and device) and 16 concepts including data flow, control flow, data creation, file manipulation, data comparison, and structure of data/file/function. The 16 concepts (10 verb type concepts and 6 adjective type concepts) are shown in Table 1.

Table 1: Concepts provided by X-JRDL.

Concept	Meaning
DFLOW	Data flow
CFLOW	Control flow
ANDSUB	And-tree structure
ORSUB	Or-tree structure
GEN	Data creation
RET	Retrieve a record in a file
UPDATE	Update a record in a file
DEL	Delete a record in file
INS	Insert a record in a file
MANIP	File manipulation
EQ, NE, LT, GT, LE, GE	Logical operators

There are several verbs to represent one of these concepts. For example, to specify a concept *data flow*, we can use i*nput, output, print out, display, and send*, and so on. Each concept has its own case structure. The "cases" (Fillmore 1968) mean concept about agents, objects, goals of the operations (Shank 1977). For example, the *data flow* (DFLOW) concept has objec*t, source*, *goal*, and *instrument* cases. The object case object corresponds to a data which is transferred from the source case object to the goal case object. So, a noun assigned to the object case should be a data type noun. A noun in the source or goal cases should be either a human or a function type noun. If and only if a human type noun is assigned to source or goal cases, some device type noun should be specified as an instrument case. These are illustrated in Fig. 1.



case	noun type
object	data
source	function, human
goal	function, human
instrument	device

Figure 1: Case structure of data flow (DFLOW).

When a user wants to write requirements of another application domain, he may need a verb not categorized into these 16 concepts. In such a case, he can use a new verb if he defines its case structure.

Since a newly defined verb, its concept, and its case structure can be registered in the verb dictionary, he can use his own verbs as well as provided verbs.

The case structure of each verb enables to detect illegal usages of data and lack of cases. Suppose a requirement sentence, *"A user enters a retrieval command with a terminal."* Since the objective is "*a retrieval command"* that is data type noun, *"enters"* should be categorized into the DFLOW concept. With the case structure of the DFLOW, this sentence will be analyzed as shown in Table 2.

Table 2: Analysis of a requirement sentence *"A user enters a retrieval command with a terminal."*

Concept: DFLOW

object	source	goal	instrument
retrieval command	user	*NOT specified*	terminal

In this sentence the goal case noun is not specified. If indispensable case is not specified, previously specified nouns of the same type become candidates of the omitted case. In this way, a requirement sentence is transformed into an internal representation named CRD (Conceptual Requirements Description). CRD is exactly based on the case structures.

X-JRDL provides to use pronouns and omission of nouns. We frequently come across such features in Japanese sentences. The X-JRDL analyzer automatically assigns a concrete word into a pronoun or a lacked case.

The X-JRDL analyzer has a dictionary of nouns, verbs and adjectives. When a requirements definer uses a word which is not appeared in the dictionary, the analyzer guesses a type of new noun and a concept of new verb and adjective with the Requirements Frame (Ohnishi 1996).

3 TRANSFORMATION OF SRS ORGANIZATION

Specific requirements in SRS can be organized differently and there exist several templates for SRS (IEEE 1998). The section 3.2 of these templates are different, but other sections are same each other. In other words, only functional requirements are differently specified.

As shown in section 2, each requirement sentence with X-JRDL can be transformed into CRD representation. We can express same requirement sentence differently. For example, "*User inputs commands to system with keyboard*" can be expressed as "*System accepts commands from user with keyboard*" or "*Commands form user with keyboard are received by system*." These three sentences can be transformed into the same CRD representation. This fact means that the CRD representation is independent of surface expression and shows conceptual meaning. We can derive surface expression depending on organizations from CRD representation.

For the organization of user class, we focus on human type nouns in an SRS with X-JRDL, because these nouns are external users of the SRS. Then we make sections for each of the nouns.

For the organization of stimulus, we focus on requirements sentences of data flow. If a human type noun is assigned as the source case object, a noun assigned as the object case of the sentence can be regarded as an external input. Then we make sections for each of the external inputs.

For the organization of functional hierarchy, we focus on function type nouns. Then we make sections for each of the nouns. Figure 2 shows a part of SRS of stock management system organized by functional hierarchy.

1) Stock management system consists of stock-in manager, stock-out manager, and stock file manager.
1.1) Stock-in manager
It receives a control from stock managing officer and also gets stock-in data including item name and amount via keyboard, then with them updates stock master file and returns the control to stock managing officer.
1.2) Stock-out manager
It consists of retriever, stock-deliverer, and orderer.
1.2.1) Retriever
It receives control from stock managing officer.
....

Figure 2: SRS organized by functional hierarchy.

X-JRDL analyzer first clarifies unknown words by asking to describer, and then divides complex sentences and compound sentences into simple sentences each of which has just one verb. Next, it transforms each simple sentence into internal representation, namely CRD representation. Table 3 shows transformed eight CRD representations of the SRS shown in Figure 2.

In order to transform into SRS of user class organization, we focus on human type nouns. There exists only one human type noun in the example.

Table 3: CRD representations.

1: concept	ANDSUB
agent	stock management system
object	stock-in manager
object	stock-out manager
object	stock file manager
2: concept	CFLOW
source	stock managing officer
goal	stock-in manager
3: concept	ANDSUB
agent	stock-in data
object	item name
object	amount
4: concept	DFLOW
agent	stock-in data
source	stock managing officer
goal	stock-in manager
instrument	keyboard
5: concept	UPDATE
agent	stock-in manager
source	item name
goal	stock master file
6: concept	CFLOW
source	stock-in manager
goal	stock managing officer
7: concept	ANDSUB
agent	stock-out manager
object	retriever
object	stock-deliverer
object	orderer
8: concept	CFLOW
source	stock managing officer
goal	retriever

That is stock managing officer. In Table 3, there exist 4 CRD representations including "stock management officer." So, we can select these 4 representations and transform them into 4 requirements sentences whose subjects are stock managing officer as shown in Figure 3.

1) Stock managing officer
Stock managing officer passes control to stock-in manager and retriever. He enters stock-in data to stock-in manager with keyboard. He gets control from stock-in manager.

Figure 3: SRS organized by user-class.

In the above example, requirements including stock managing officer are specified and other requirements such as functional structures are omitted. So, each class of the users can easily confirm the correctness of transformed requirements. Other requirements will be attached to the transformed requirements in order to keep the completeness of the SRS.

In order to transform into SRS of stimulus organization, we focus on external inputs in the SRS. There exists only one external input, that is, "stock-in data." There exist two representations including "stock in data." We can select the 3rd representation and the 4th representation in Table 3 and transform them into two requirements sentences whose subjects are stock-in data as shown in Figure 4.

1) Stock-in data
Stock-in data consists of item-name and amount. Stock-in data is transferred from stock managing officer with keyboard to stock-in manager.

Figure 4: SRS organized by external inputs.

In Figure 4, only external input is specified and it is very easy to check what kind of functions receives these inputs.

We have developed a prototype system based on our method with Java. Although our system is based on Japanese-base language, our method is independent of Japanese language.

We applied our method to an SRS of stock management system using the prototype system This SRS consists of 27 requirements sentences with X-JRDL and organized by functional hierarchy. We transformed this SRS into SRS by user class and SRS by stimulus correctly, but some requirements such as functional hierarchy are not transformed, because requirements of functional hierarchy are not categorized into user-class requirements or stimulus requirements.

4 RELATED WORKS

In (Nuseibeh, Kramer, and Finkelstein, 1994), they claim that SRS is specified from multiple viewpoints and propose a consistency check method among requirements form different viewpoints.

In (Martinez, Arias, Vilas, 2005), they propose a merging method of requirements described by different stakeholders.

In (Heitmeyer, Jeffords, Labaw, 1996), authors propose a consistency checking method of formally specified SRS. The above three methods cannot generate SRS of differently organized.

In the author's previous work (Zhang and Ohnishi, 2004), a scenario from a certain viewpoint can be transformed into a scenario from a different viewpoint. The previous method enables to transform scenarios from different viewpoints, but it cannot be applied to SRS transformation.

5 CONCLUSIONS

We have developed a transformation method for SRS by different organizations. We can transform SRSs by user-class, functional hierarchy, and stimulus organizations. Automatic transformation of SRSs will contribute to generate differently organized SRSs of high quality and lessen efforts of specifying differently organized SRSs.

In (IEEE 1998) more types of organizations are specified. These are organization by mode, organization by object, organization by feature. The enhancement of the transformation method for SRS by these three organizations is left as a future work.

REFERENCES

Fillmore C.J., 1968. The Case for Case, Universals in Linguistic Theory, ed. Bach & Harrms, Holy, Richard and Winston Publishing, Chicago.

Heitmeyer C., Jeffords R., Labaw B., 1996. Automated Consistency Checking of Requirements Specifications, ACM Transactions on Software Engineering and Methodology (TOSEM), Vol.5, Issue 3, pp.231-261.

IEEE, 1998. IEEE Recommended Practice for Software Requirements Specification, IEEE std 830-1998.

Martinez A., Arias J., Vilas A., 2005. Merging Requirements Views with Incompleteness and Inconsistency, IEEE Australian Software Engineering Conference (ASWEC'05), pp. 58-67.

Nuseibeh B., Kramer J., and Finkelstein A., 1994. A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification, IEEE Transactions on Software Engineering, Vol.20, No.4, pp.760-763.

Ohnishi A., 1996. Software Requirements Specification Database based on Requirements Frame Model, IEEE 2nd International Conference on Requirements Engineering (ICRE96), pp.221-228.

Shank R., 1997. Representation and Understanding of Text, *Machine Intelligence 8,* Ellis Honrood Ltd., Cambridge, pp.575-607.

Zhang H., Ohnishi A., 2004. A Transformation Method of Scenarios from Different Viewpoints, IEEE 11th Asia-Pacific Software Engineering Conference (APSEC2004), pp.492-501.