A Comparison of Two Model Transformation Frameworks for Multiple-viewed Software Requirements Acquisition

Bingyang Wei
Department of Computer Science
Midwestern State University
Wichita Fall, Texas, 76308
Email: bingyang.wei@mwsu.edu

***Abstract*—Multiple-viewed requirements modeling allows modelers to elicit the requirements of a system from different viewpoints. Requirements are then organized and encoded in different analysis models which collaboratively form an overall understanding of the system. Model transformations among those analysis models at this stage can be used as a way to acquire requirements knowledge, thus making the set of models complete and consistent. Two frameworks are found to support such requirements acquisition: the pairwise framework and the common representation framework. In practical applications, various factors need to be considered when requirements modelers choose between the two frameworks in order to acquire requirements by analysis model transformations. In this paper, we propose a set of criteria which provides a theoretical basis for comparing the two frameworks for their effectiveness of generating models and acquiring requirements in the context of multiple-viewed requirements modeling. The results of the comparison is then presented.**

I. Introduction

Properly eliciting and modeling the requirements are important to successful software development. Multiple-viewed requirements modeling is commonly used so that a system is observed from different viewpoints by different people, and the requirements are expressed in different types of analysis models. The creation of different types of analysis models for a system is parallel and iterative by nature and might be conducted by different requirements modelers. However, it is difficult for a modeler to know whether an analysis model is complete or what requirements are missing from the current version of a model [6]. The modelers need to be made aware of the missing requirements of analysis models. In order to make explicit the missing requirements in an analysis model, Delugach proposed the idea of conceptual feedback [4] which can provide prompts for the missing requirements to modelers (see Figure 1). This approach is based on the requirements knowledge overlap among analysis models of the same system. During requirements analysis stage, already-created analysis models ($Model_0$ to $Model_n$ in (b) of Figure 1) are transformed to a target analysis model $Model_x$. This process introduces new requirements to $Model_x$, which make it more complete, and more importantly, reveal some requirements acquisition opportunities to the modeler of $Model_x$.

DOI reference number: 10.18293/SEKE2017-070

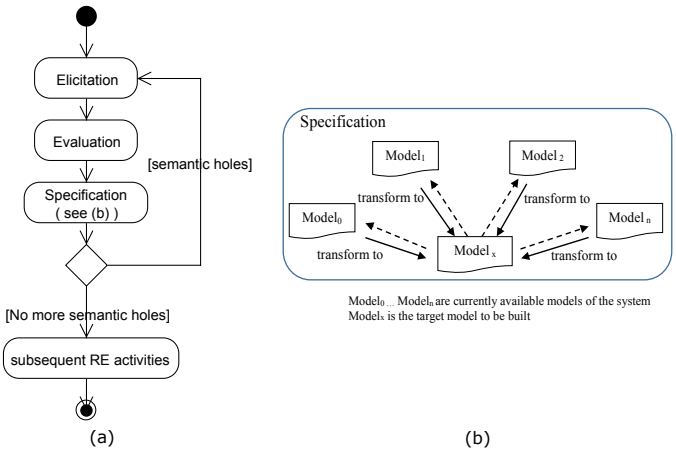

Fig. 1. Conceptual feedback in RE process

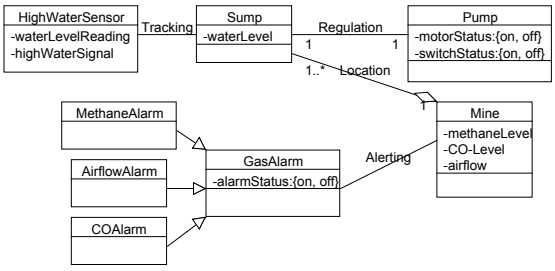

Fig. 2. Original class diagram before model transformations

A simple example of adopting the conceptual feedback approach to get more requirements and reveal requirements acquisition opportunities in a UML class diagram is shown in Figure 2 and Figure 3. Through model transformations, the original UML class diagram shown in Figure 2 is augmented by requirements from the current state diagrams and sequence diagrams of a Mine Safety Control system [10]. The resulting augmented class diagram is shown in Figure 3. Grey classes represent new classes that are added in through

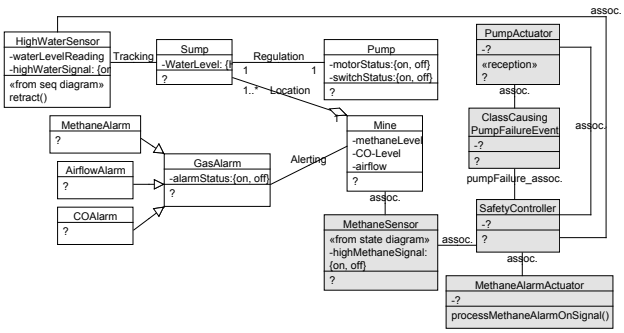

Fig. 3. Augmented class diagram after model transformations

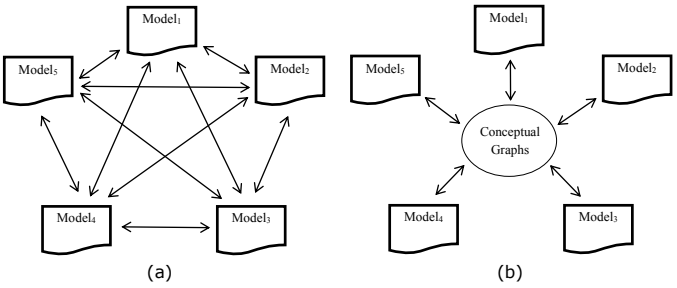

Fig. 4. PWF and CRF

model transformation. Note that the augmented class diagram contains question marks which denote the potentially missing requirements. In other words, they are requirements of the class diagram which cannot be automatically generated based on the existing requirements knowledge in other currently available models, therefore need to be explicitly provided by modelers. For example, class **`PumpActuator`** in Figure 3 has two question marks: an unspecified attribute and an unspecified operation.

Because of the presence of requirements acquisition opportunities, the augmented analysis model is considered as incomplete so modelers are invited to complete it by providing the missing requirements. This enables additional new requirements to be elicited by fixing those question marks (This process is shown as a feedback arrow from Specification to Elicitation in (a) of Figure 1). After that, the augmented model with question marks resolved would in turn affect other models (dotted arrows in (b) of Figure 1), causing further generation and completion processes in other analysis models. The process may repeat until no more new requirements knowledge can be acquired by transforming models, i.e., the set of models is internally complete and self-consistent.

There are two common used frameworks in Requirements Engineering community that can support software requirements acquisition by model transformations. They are the pairwise framework (PWF) and the common representation framework (CRF) shown in Figure 4. The former framework supports requirements acquisition by direct transformations between models whereas the latter by relying on a common knowledge representation that describes the semantics of the analysis models.

For PWF, transformations between each pair of UML diagrams have been studied by many researchers [5][7][8], and they collaboratively carried out a well-understood semantic description of model transformations in PWF. A summary of transformation rules between each pair of three UML diagrams is available in [11]. As for CRF, Wei and Delugach [13] proposed using conceptual graphs (CGs) [9] as the common representation in CRF to transform and acquire requirements for analysis models. A detailed description of model transformations with CGs in the CRF used for requirements acquisition is provided in their work [11][14][12].

An important research problem is that the two frameworks' effectiveness in transforming models and acquiring requirements knowledge is largely unknown. There is no theoretical basis for evaluating which framework is more effective in transforming models and acquiring requirements in the context of multiple-viewed requirements modeling. A detailed analysis of the capability of eliciting requirements from requirements modelers, the capability of preserving semantics and the extensibility of each framework, and an objective comparison is needed so that researchers and modelers can rely on this proposed criteria and comparison to choose between frameworks for addressing requirements acquisition problem.

The remainder of the paper is organized as follows: In Section 2, we explain the approach we use in order to compare the two frameworks for their effectiveness of exposing requirements acquisition opportunities and propose the evaluation criteria. In Section 3, the two frameworks are evaluated according to a set of criteria; the results of this comparison are presented. Section 4 discusses the limitations of this work and concludes the paper.

II. Comparison Methodology

In this section, details of the comparison are presented. These include the analysis models used in both frameworks, comparison procedures, software system examples and proposed criteria.

In this work, three types of UML diagrams are considered for both frameworks (Figure 5). They are class diagrams, state diagrams, and sequence diagrams which are among the most commonly used diagrams in UML for specifying an object-oriented system. The reason for choosing them is that the structure, state, and interaction views of a system provide a sufficiently broad range of the semantics of object-oriented models to show the generality of our approach. The conversions and comparisons are done by the author: conversion rules for both frameworks are derived based on the latest UML specification [1], and the conversions are conducted by strictly following the rules. We tried to limit the bias to the minimum, so it does not threat the validity of the results.

The way to conduct our comparison is shown in Table I. One model is considered as the target model, and two other models are transformed to it using one of the two frameworks: In PWF, two models are transformed to the third one directly, while in CRF, two models are first converted to CGs from which

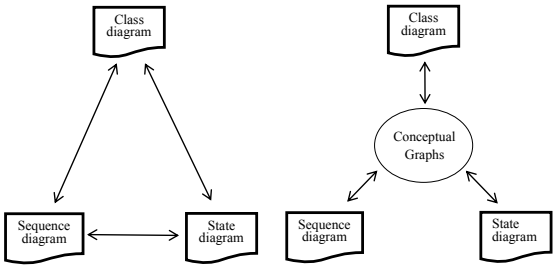

Fig. 5. Three UML diagrams in two frameworks

TABLE I
TRANSFORMATION STRATEGY

Transf. #	Source models	Target model
1	state diag., sequence diag.	class diag.
2	sequence diag., class diag.	state diag.
3	state diag., class diag.	sequence diag.

the third diagram is derived. The target models generated by different frameworks are then compared based on a set of criteria.

Three non-trivial software case studies are used. They are the University Information System (UnivSys.)[2], which is an information system; the Cryptanalysis System (Cryptanlys.)[3], which is an AI system, and the Mine Safety Control System (MineSys.)[10], which is a real time control system. The use of three case studies in three different domains makes sure that our comparison results are not dependent on the domain. The results of working out the three case studies in both frameworks are available in the appendices of [11].

A set of criteria that can be used to evaluate frameworks used for requirements knowledge acquisition is then proposed. Although only two specific frameworks are compared in this paper, these criteria are not limited to them. The criteria are meant to be applicable to any transformation framework that claims to address the requirements knowledge acquisition problem. The set of criteria is presented here:

1) Capability of acquiring missing requirements
2) Capability of generating definite requirements
3) Percentage of the missing requirements in generated UML diagrams
4) Extensibility
5) Knowledge acquisition effort

The reason these criteria are picked is the purpose of the model transformations in the two different frameworks: we are interested in the fact that which framework will produce more requirements acquisition opportunities for a modeler.

III. COMPARISON RESULTS

The results of the comparison are shown here. Each of criteria proposed in last section is explained in detail.

A. Capability of Acquiring Missing Requirements

Both model transformation frameworks can generate a target UML diagram and reveal requirements acquisition opportunities. In this work, a requirements acquisition opportunity refers to something that needs clarification in an augmented UML diagram. For example, a question mark on an association in a class diagram or an automatically generated class name like **`ClassCausingPumpFailureEvent`** in Figure 3. It reveals some missing requirement that a modeler needs to provide, and there is no way that a framework can generate that missing requirement automatically. Since the presence of requirements acquisition opportunities indicates the possible missing requirements knowledge, a high number of requirements acquisition opportunities in a UML diagram is a sign that more requirements knowledge will be potentially acquired from a modeler. This criterion evaluates the capability that a framework has to acquire the missing requirements. This is measured by counting the number of requirements acquisition opportunities in the UML diagrams generated by a framework. An advantage of using the quantity of requirements acquisition opportunities as the metric is that this only depends on the representation of UML diagrams and not on any subjective judgment of incompleteness or experience of requirements modelers. For example, different class diagram modelers will have different ways to complete the same augmented class diagram (Figure 3). So instead of asking several modelers to really fill in the holes and getting an average, we simply count the number of requirements acquisition opportunities that need to be filled, since these requirements (requirements acquisition opportunities) are missing for sure, and need to be provided by the requirements modelers.

During the comparison, given the same state and sequence diagrams of a software system, two class diagrams are generated by PWF and CRF, respectively; then the number of requirements acquisition opportunities yielded in each of the two generated class diagrams is counted. The types of requirements acquisition opportunities that we are counting in an augmented class diagram are listed under the gray row "In generated class diagram" in Table II. The same comparison process works for augmented state diagrams and augmented sequence diagrams. The types of requirements acquisition opportunities that we are counting in a generated state diagram are listed under the gray row "In generated state diagram" and the types of requirements acquisition opportunities that we are counting in a generated sequence diagram are listed under the gray row "In generated sequence diagram" in Table II. The complete results of evaluating the generated class, state and sequence diagrams for three case studies in the two frameworks are listed in Table II. Higher values are better. Based on the result, CRF generated more missing requirements than PWF did.

B. Capability of Generating Definite Requirements

In a generated target UML diagram, besides requirements acquisition opportunities, there are newly generated requirements that are certain. We named these requirements "definite inferred requirements" in this paper, and they do not need

TABLE II
COMPARISON OF THE CAPABILITY OF ACQUIRING MISSING REQUIREMENTS IN TWO FRAMEWORKS

	PWF			CRF		
requirements acquisition opportunities	UnivSys.	Cryptanlys.	MineSys.	UnivSys.	Cryptanlys.	MineSys.
In generated class diagram						
Number of unknown class names	7	8	4	7	8	6
Number of unknown attribute names	0	0	0	15	20	25
Number of unknown operation names	13	14	27	49	59	81
Number of unknown association names	12	11	13	12	11	15
Total	32	33	44	83	98	127
In generated state diagram						
Number of unknown/potential states	20	18	19	20	18	13
Number of unknown transitions	4	3	5	5	4	3
Number of unknown events	5	3	2	5	4	3
Number of unknown effects	0	0	0	7	7	5
Number of unknown guards	0	0	0	15	14	10
Number of unknown entry/exit, do activities	0	0	0	60	54	39
Number of state invariants	0	0	0	10	10	7
Total	29	24	26	122	112	80
In generated sequence diagram						
Number of unknown neighboring lifelines	6	8	8	6	7	8
Number of unknown messages	0	1	0	0	0	0
Number of unknown execution specs	0	0	0	25	37	58
Total	6	9	8	31	44	66

further clarification by modelers. For example, the method **`processMethaneAlarmOnSignal()`** in the augmented class diagram in Figure 3. This criterion evaluates the capability that a framework has to acquire the definite requirements knowledge. This is measured by counting the number of definite model elements in the UML diagrams generated by a framework. The types of definite requirements that we are counting in the three generated UML diagrams are listed in the first column in Table III. The results of evaluating the generated class, state and sequence diagrams for three case studies in two frameworks are listed in Table III. Higher values are better. Based on the result, the difference between PWF and CRF is small.

C. Percentage of the Missing Requirements in Generated UML Diagrams

This criterion evaluates the incompleteness of the UML diagrams generated by PWF and CRF. The incompleteness of a generated UML diagram is the number of requirements acquisition opportunities over the number of overall generated model elements in a generated UML diagram. The results of evaluating the incompleteness of generated class, state, and sequence diagrams for three case studies in two frameworks are listed in Table IV. For example, for the generated class diagram of the UnivSys. by CRF, it is 83.8% incomplete and 16.2% complete. Since the purpose of this paper is to evaluate requirements acquisition, higher percentages are better. Based on the result, CRF produces more incomplete diagram than PWF.

D. Extensibility

In this work, three types of UML diagrams are chosen for both frameworks. However, more UML diagrams can be added to expand the frameworks. This criterion evaluates the extensibility of a framework by measuring the amount of effort needed to introduce another type of UML diagram (activity diagram in this case) in both frameworks. For simplicity, only limited model elements in activity diagrams are considered: Activity partitions, Activity nodes, control flows, Forking, and joining. When a new type of model is introduced, new transformation rules need to be developed for both frameworks. We are counting the number of new rules developed for a framework to evaluate its extensibility. Lower values are better. In PWF, 26 new rules are needed to add a fourth diagram (the activity diagram), while in CRF, only 4 new rules are needed. Apparently, CRF is more extensible than PWF.

TABLE III
COMPARISON OF THE CAPABILITY OF GENERATING DEFINITE REQUIREMENTS IN TWO FRAMEWORKS

	PWF			CRF		
Definite requirements	UnivSys.	Cryptanlys.	MineSys.	UnivSys.	Cryptanlys.	MineSys.
In generated class diagram						
Number of definite classes acquired	5	4	8	5	4	8
Number of definite attributes acquired	2	3	11	1	2	5
Number of definite operations acquired	11	9	0	10	9	0
Number of definite association names acquired	0	0	0	0	0	0
Total	18	16	19	16	15	13
In generated state diagram						
Number of definite state machines	5	4	5	5	4	5
Number of definite states	0	0	0	0	0	0
Number of definite transitions	15	14	9	10	10	7
Number of definite events	10	8	7	10	10	7
Number of definite effects	9	9	7	9	9	5
Total	39	35	28	34	33	22
In generated sequence diagram						
Number of definite sequences	4	2	4	9	9	11
Number of definite messages	6	6	6	6	6	6
Number of definite execution specs	5	4	0	5	3	0
Number of definite states	6	10	14	6	10	14
Number of definite combined fragments	2	2	10	0	0	0
Total	23	24	34	26	28	31

TABLE IV
COMPARISON OF INCOMPLETENESS OF GENERATED UML DIAGRAMS BY TWO FRAMEWORKS

Generated	PWF			CRF		
UML diagrams	UnivSys.	Cryptanlys.	MineSys.	UnivSys.	Cryptanlys.	MineSys.
Class diagram	64.0%	42.7%	20.7%	83.8%	78.2%	54.4%
State diagram	67.4%	40.7%	27.3%	86.7%	77.2%	61.1%
Sequence diagram	69.8%	48.2%	19.1%	90.7%	78.4%	68.0%

E. Knowledge Acquisition Effort

This criterion is used to evaluate and compare the amount of effort needed to complete requirements acquisition opportunities in the UML diagrams generated by PWF and CRF. When a modeler is presented with UML diagrams with requirements acquisition opportunities, she needs to look at each hole and choose to either fill it in or delete it. So, in this work, the knowledge acquisition effort is measured by counting the number of requirements acquisition opportunities that needed to be considered (Table V). Higher values are better. Based on the result, CRF requires more effort than PWF.

IV. DISCUSSION OF THE RESULTS

Criterion 1 compared the capability of acquiring missing requirements of the two frameworks. Based on the results of applying both frameworks to three case studies, the CRF outnumbered the PWF in all generated UML diagrams in three case studies (see Table II). In other words, using CRF, more possible requirements may be acquired from a modeler in the multiple-viewed requirements modeling context. Currently, there is no metric to measure the usefulness of a requirements acquisition opportunity. Criterion 2 compared the capability of generating definite requirements in the two frameworks.

TABLE V
COMPARISON OF THE KNOWLEDGE ACQUISITION EFFORT NEEDED IN UML DIAGRAMS GENERATED BY TWO FRAMEWORKS

Generated	**PWF**			**CRF**		
UML diagrams	UnivSys.	Cryptanlys.	MineSys.	UnivSys.	Cryptanlys.	MineSys.
Class diagram	32	33	44	83	98	127
State diagram	29	24	26	122	112	80
Sequence diagram	6	9	8	31	44	66

Based on the results of applying both frameworks to three case studies, PWF outnumbered CRF in almost all generated UML diagrams in three case studies. In other words, using PWF, more determined requirements can be generated. This implies that PWF is better at generating target models from source models. However, the difference is not much. Given a set of requirements acquisition opportunities and a set of definite requirements in a generated target UML diagram, readers may wonder which one is more desired. In this work, we are aiming for knowledge acquisition, so more requirements acquisition opportunities are desired.

Built upon the previous two criteria, the comparison based on criterion 3 shows that UML diagrams generated in CRF are generally more incomplete than in PWF. This matches our first conclusion that CRF is more capable of exposing requirements acquisition opportunities in the multiple-viewed requirements modeling context.

Criterion 4 is crucial for framework developers since a framework grows when new types of diagrams are introduced. By introducing a new type of UML diagram, activity diagrams, in both frameworks, CRF is found to be clearly more extensible than PWF. This advantage becomes more evident when a framework supports more UML diagrams.

More requirements acquisition opportunities mean more effort involved in resolving them. In CRF, a modeler has to go through each requirements acquisition opportunity to decide if it is indeed useful and meaningful. In this work, we assume that every requirements acquisition opportunity takes the same amount of time to be completed. We conclude that it takes more time to complete a UML diagram generated by CRF than by PWF because more need to be taken into account.

Besides the purpose of requirements acquisition, CRF has the advantage of reasoning with the CGs Reservoir (the central conceptual graphs in Figure 5) which stores the entire requirements of the system. We recommend CRF as a better model transformation framework for the purpose of requirements acquisition.

Two current limitations are the lack of automation support and usage of simple case studies. Future work obviously will focus on automation and industrial examples. The current work assumes its input UML diagrams are syntactically and semantically correct. If errors exist, those errors will be converted to CGs or other UML diagrams and then used to infer incorrect or badly formed requirements knowledge. Retracting the CGs inferred from this wrong knowledge is a time-consuming and complex task that the current framework cannot handle.

V. CONCLUSION

In this paper, we proposed and conducted a comparison of two model transformation frameworks that can be used in requirements acquisition. Five general comparison criteria are provided for evaluating the frameworks so that modelers can reference them when choosing frameworks. Based on the results, common representation framework exceeds pairwise Framework in term of requirements acquisition.

REFERENCES

[1] OMG Unified Modeling Language™ (OMG UML), Version 2.5. object management group, 2015.

[2] S. W. Ambler. *The object primer: Agile model-driven development with UML 2.0*. Cambridge University Press, 2004.

[3] G. Booch, R. Maksimchuk, M. Engle, B. Young, J. Conallen, and K. Houston. *Object-oriented Analysis and Design with Applications, Third Edition*. Addison-Wesley Professional, third edition, 2007.

[4] H. S. Delugach. An approach to conceptual feedback in multiple viewed software requirements modeling. In *Joint proceedings of the second international software architecture workshop (ISAW-2) and international workshop on multiple perspectives in software development (Viewpoints' 96) on SIGSOFT'96 workshops*, pages 242–246. ACM, 1996.

[5] A. F. Egyed. *Heterogeneous view integration and its automation*. PhD thesis, University of Southern California, 2000.

[6] D. Firesmith. Are your requirements complete? *Journal of Object Technology*, 4(1):27–44, 2005.

[7] B. Nuseibeh, J. Kramer, and A. Finkelstein. A framework for expressing the relationships between multiple views in requirements specification. *Software Engineering, IEEE Transactions on*, 20(10):760–773, 1994.

[8] P. Selonen, K. Koskimies, and M. Sakkinen. Transformation between uml diagrams. *Journal of Database Management*, 14(3):37–55, 2003.

[9] J. F. Sowa. Conceptual structures: information processing in mind and machine. 1983.

[10] A. Van Lamsweerde et al. Requirements engineering: from system goals to uml models to software specifications. 2009.

[11] B. Wei. *A comparison of two frameworks for multiple-viewed software requirements acquisition*. PhD thesis, Ph. D. thesis, University of Alabama in Huntsville, 2015.

[12] B. Wei and H. S. Delugach. A framework for requirements knowledge acquisition using uml and conceptual graphs. In *Software Engineering Research, Management and Applications*, pages 49–63. Springer, 2016.

[13] B. Wei and H. S. Delugach. Transforming uml models to and from conceptual graphs to identify missing requirements. In *International Conference on Conceptual Structures*, pages 72–79. Springer, 2016.

[14] B. Wei, H. S. Delugach, E. Colmenares, and C. Stringfellow. A conceptual graphs framework for teaching uml model-based requirements acquisition. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 71–75. IEEE, 2016.