Research on reverse engineering from formal models to UML models

Wei Yan[1] , Yugen Du[1]
[1]*Software Engineering Institute*
East China Normal University
ShangHai, china
Email:{vito.yanw, ygdu}@sei.ecnu.edu.cn

***Abstract*—The Unified Modeling Language (UML) provides a graphical notation to express the design of object-oriented software systems and has become the de facto industry standard for software design. However UML lacks precise semantics and is semi-formal. Formal specification languages are intended to provide precise and complete models for proposed software systems. Many researchers have done a lot of work in translating UML models into formal models to validate UML models. But in this paper, we discuss the reverse engineering problem, that is, when the formal models are validated and corrected, how to reverse them to UML models. We think this problem is more meaningful for software engineer. This paper presents a method that translates formal models into UML models by XMI and its Schema, and then testifies the feasibility and correctness of the reverse method by Unifying Theories of Programming (UTP) [1].**

***Keywords*-UML; formal method; reverse transformation; UTP**

I. INTRODUCTION

As a standard modeling language provided by OMG, the Unified Modeling Language (UML) has been widely used in industry [2]. However, UML lacks strict and precise semantics, it is inevitably ambiguous, incomplete, inconsistent [3]. Formal method, as a mathematical language, is capable of rigorous analysis and reasoning. Formal modeling can be a good solution to some of the problems and shortcomings about UML, but is not visual or easy to understand because the language is purely a formal mathematical language. For most software developers of the industrial sector who lack mathematical training, it is difficult to model using formal methods.

In order to combine the advantages of formal methods and UML, many researchers try to give a precise formal semantics to UML, translate UML models into formal model described with formal method, eliminate UML models uncertainty, ambiguity and incomplete through the analysis and reasoning of formal models. But most of translation is one-way, which is just transformation from UML models into formal models without reverse transformation from formal models into UML models. This "going back and forth"between the realms of informal and formal specification is necessary since it is impossible to verify whether an informal requirement is equivalent to a formal description [4].Therefore it is necessary to provide a method of reversely transforming formal models to UML models to eliminate the defects contained in UML models by reasoning and validating formal model. Software developers can not only get the benefits of using formal mathematical reasoning to prove the model, but also get the benefits of visual modeling using UML.

XMI [5], which has a good scalability , is a standard method of model transformation provided by OMG. Using XMI as a vehicle of model transformation, the paper provides a transformation method to re-transform formal models to UML models.

In section 2, we analyses the existing transformations and the reverse transformations. In section 3, we present a indirect reverse transformation. In section 4, we testify the feasibility and correctness of the reverse method by Unifying Theories of Programming (UTP) [1] proposed by Hoare and He. In section 5, we will conclude and discuss further work.

II. ANALYSES THE TRANSFORMATIONS AND THE REVERSE TRANSFORMATIONS

Formal methods can precisely describe the property of system in mathematic language and provide a systematical way to describe, develop and validate the system framework [6, 7]. Formal models based on formal method are precise, analytical.

UML, as a visual modeling language, lacks rigorous, precise semantics. Research communities have been working on formalization of UML for a decade. Formalization of UML can be divided into two kinds of approaches.

a)*:* The first kind of approach is transforming UML models to Formal models.

The formal models based on formal method can be reasoned with the existing tools. In the past decade, there are a lot of publications about this approach. [8–21] have proposed a lot of methods to translate formal models to UML models.

b)*:* The second kind of approach is giving an abstract syntax to UML diagrams and providing a formal semantics for the UML. [22–35] have provided formal semantics for the UML.

Using the two kinds of approaches, we can realize the formalization of UML. But the existing publications mainly focus on the advantages of the strict mathematical reasoning

978-0-7695-4312-3/10 $26.00 © 2010 IEEE
DOI 10.1109/PAAP.2010.54

and analyzing while do not care about how to reversely transform formal models to UML models which can be refined into the next phase UML models. [22–24] have proposed a reverse transformation, but it will lead to the loss of information. After the initial UML model is transformed and reversely transformed, the result UML model will lose some information of the initial one even if we have not modified the formal model transformed from the initial UML model.

In different levels, some semantics of UML is different. It is impossible to verify whether a UML model is equivalent to a formal model, therefore it is necessary to explore a reverse transformation.

For Safety and reliability, the developer [36–38] adopts formal method to develop directly the security-concern system. [39–44] proposed some techniques that can derivate UML Diagrams from formal models to help those who is not familiar with formal methods understand the formal specification. But those techniques using UML diagrams are proposed for the graphical representation of formal specification developed directly by the formal model developer. The result UML diagrams are too incomplete and complicated to be refined into the next phase UML models.

To avoid the loss of information and refine the result UML model, we will present an indirect reverse transformation in section 3.

III. Method of Reverse Transformation Model

In this section we introduce the reverse transformation method by the vehicle of XMI to solve the problem mentioned in section 2.

XMI (XML Metadata Interchange) [5] is a format standard for the exchange of metadata and meta-model provided by OMG, which integrates three industry-standard XML [45], UML [46] and MOF [47].

XMI includes two generation rules. One is the XMI schema document generation rules; the other is the XMI document generation rules. XMI Schema document is used to verify the XMI document content. Two rules of XMI support UML and MOF. Therefore, using XMI generation rules, we can produce XMI Schema documents and XMI documents from the MOF and UML. Because the UML metamodel is an instance of the MOF model and UML is defined by the MOF [48], XMI Schema documents generated by the UML metamodel can be used to verify the XMI document generated by UML models. XMI also provides a mechanism for the exchange of the UML model changes. It does not have to transmit the entire model information at every exchange time thus reduces the costs of information exchange [5].

Suppose N stands for a new XMI document mapped from the new UML model, O for an old XMI document mapped from the old model and D for a XMI document of the difference between O and N information. D is encoded by XMI Differential elements.

Generally, there are three operations — deletion, addition and replace on the elements of UML model, corresponding to the three Differential Encoding elements — **delete**, **add**, **replace** in XMI document.

If we want to achieve the UML model updating, we only need to combine the XMI document of D with O in defined order to form the XMI documents of N. The equation $N = O + D$ shows that O combined with D can form a new XMI document N of the updated model. The UML model included in N is the new UML model and also is updated from the old UML model O.

If formal model needs to be corrected **n** times to eliminate the defects of model, we will get a difference XMI document sequence $D1, D2, \ldots\ Dn$, each $Di(i = 1, 2, \ldots, n)$ is called a XMI difference encoding element for the modification of UML model sequentially. Integrating the difference XMI document sequence $D1, D2, \ldots, Dn$ into original XMI document O in order will generate a new XMI document. The UML model mapped from the new XMI document is consistent with the corrected formal model in semantics.

According to the analysis in section 2, the existing techniques will lead to the loss of information and the shortage that the result UML model can not to be refined.

XMI supports the exchange of models in differential form, so we can use the XMI transmitting differences mechanism to solve the problems of the existing reverse transformation. We will present a reverse transformation combining the XMI transmitting differences mechanism below. Fig.1 shows the process of the reverse transformation.

Step 1*:* we export the initial XMI document from the initial UML model by CASE TOOL. The initial XMI document includes all the information of the initial UML model.

Step 2*:* using the existing transformation, we transform the initial UML model to the initial formal model.

Step 3*:* after reasoning and modifying the initial formal model, we will get a modified formal model.

Step 4*:* we extract the differences between the initial formal model and the modified formal model, or rather, the changes from the former to the latter. Each modification will lead to a change, or rather, a difference. Then, in the order of the analyst's modification, we organize all the differences into a sequence of differences, one of which means one modification. The sequence of differences is also a sequence of differential formal models between the initial formal model and the modified one.

Step 5*:* all the changes that occurred should be reflected by the initial UML model when the analyst modified the initial formal model, which means we should do some modifications to the initial UML model according to the modifications that the analyst have done to the initial formal model. So in this step, we should reversely transform

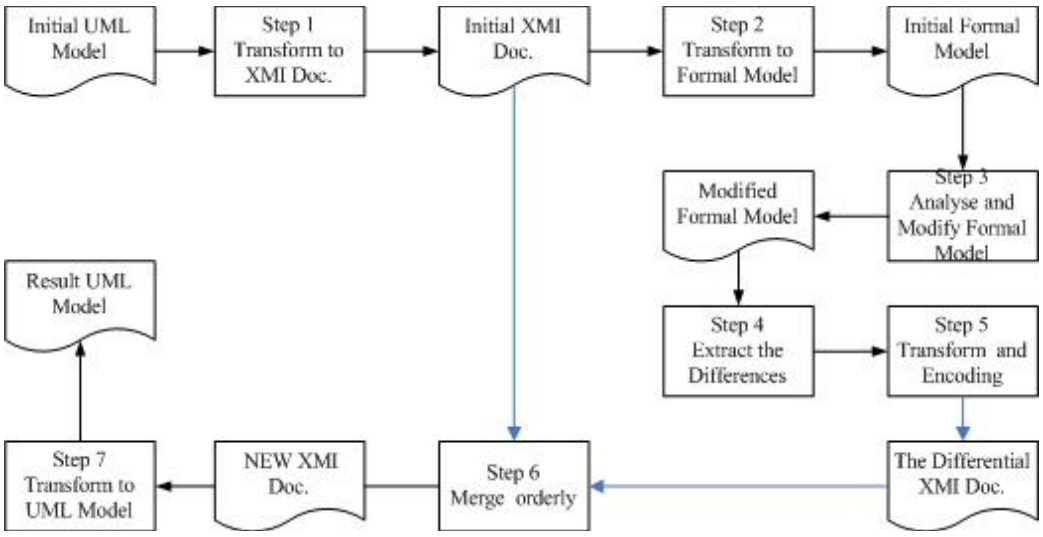

Figure 1. The Process of Reverse Transformation illustrates reverse transformation key steps.

the sequence of differential formal models to a sequence of differential UML models in order. Then we use XMI differential elements to describe the sequence of differential UML models and we will get a sequence of differential XMI documents in the order of the sequence of differential UML models. One such document includes all the information of one differential UML model in the sequence of differential UML models.

Step 6*:* we combine the initial XMI document with the sequence of differential XMI documents in order to form a new XMI document. The sequence of differential XMI documents includes all the modifications that should be orderly done to the initial UML models. So the new XMI document includes all the information of the result UML model.

Step 7*:* we can get the result UML model from the new XMI document by CASE TOOL. The result UML model is reversely transformed from the modified formal model indirectly and it is equivalent to the one refined from the initial UML model.

As we can see from step 1 to step 7, our method is an indirect reverse transformation.

Transformation from the modified formal model to the result UML model is a process from abstract to concrete. If we reversely transform the modified formal model to the result UML model directly, the result UML model will lose some information of the initial UML model. In our method, we just reversely transform the changes from the initial formal model to the modified formal model into the changes from the initial UML model to the result UML model and combine orderly the initial UML model with the sequence of differential UML models to form the result UML model. The result UML model does not lose information of the initial UML model except the information that is deleted or replaced. The result UML model can also be refined by the graphical developer.

IV. The Feasibility and Correctness of Reverse Transformation

In Section 3 we have discussed methods of how to model reverse conversion. In this section, using Galois connection proposed by Hoare and He in the UTP (Unifying Theories of Programming) [1], we will demonstrate the feasibility and correctness of the reverse transformation by the predicate-based semantic model.

Suppose set S is a UML model set in predicate space, where each predicate $P \in S$ denotes a UML model. Also assume that a set T is a formal model set in predicate space, where each predicate $Q \in T$ denotes a formal model.

S and T are sets of partial order $\supseteq$. $P1 \supseteq P2$ means that $P2$ is refined from the $P1$. The partial order relation can also use symbol $\Rightarrow$. $P1 \Rightarrow P2$ means that the model $P1$ is better than $P2$.

$F : S \rightarrow T$ is a mapping from UML models to form models.

$F^{-1} : T \rightarrow S$ is a reverse mapping of F.

For $\forall Xi \in S, \forall Yi \in T(i = 1, 2, \dots\ n), X1 \wedge X2 \dots \wedge Xn$ means the merger of differential sub-blocks of a UML model, $Y1 \wedge Y2 \dots \wedge Yn$ means the merger of differential sub-blocks of a formal model. The result of mapping after merging the sub-blocks of a model and that of merging after separately mapping them are the same. Therefore, F and F^{-1} satisfy the universal distributive law of the merger

operator $\wedge$ (See the following formulas (1) and (2)).

$$F(X1 \wedge X2 \cdots \wedge Xn) = F(X1) \wedge F(X2) \ldots \wedge F(Xn) \quad (1)$$

$$F^{-1}(Y1 \wedge Y2 \cdots \wedge Yn) = F^{-1}(Y1) \wedge F^{-1}(Y2) \ldots \wedge F^{-1}(Yn) \quad (2)$$

Mapping from UML models to formal models is a process from concrete to abstract after which some information in UML models will lose in formal models, namely, Function F maps the predicate set with a stronger descriptive ability to the one with a weaker descriptive ability. Therefore,$for \forall X \in S, X \supseteq F \circ F^{-1}(X)$ satisfies(See the formula (3)). Reverse transformation from formal models to UML models is a process from the abstract to the concrete and the information of the formal models will not be lost, that is, F^{-1} will not produce weakening effect to the predicate. Therefore, $for \forall Y \in T$ also satisfy $Y = F^{-1} \circ F(X)$ (See the formula (4)).

$$X \supseteq F^{-1} \circ F(X) \quad (3)$$

$$Y = F \circ F^{-1}(Y) \quad (4)$$

Theorem IV.1 (Galois Connection)**.** *Function pair* $(F,\ F^{-1})$ *established on the sets of* S *and* T *with a partial order* $\supseteq$ *is a Galois connection [45] if the two functions* F *and* F^{-1} *satisfy (1), (2), (3) and (4).*

Suppose a UML model $P \in S$, a formal model $Q \in T$, Q mapped from P, namely, $Q = F(P)$, Q reverse transform to the UML model P', namely, $P' = F^{-1}(Q)$. According to Galois connection's property (3), we know $P \supseteq F^{-1} \circ F(P)$, namely, $P \supseteq P'$. It denotes the UML model P' is worse than the UML model P thus after the initial formal model has been reversely and directly transformed, the result UML model will be worse than the initial one even we did not modify the initial formal model transformed from the initial UML model. As a result, direct reverse transformation from formal models to UML models is not suitable. Based on this analysis, we put forward an approach of indirect reverse transformation.

When an initial formal model needs to be modified, we regard each modification as an increment of the initial formal model. Suppose Sequence $C1, C2 \ldots, Cn$ is a incremental model sequence of the formal model Q, where each predicate $Ci(i = 1, 2, \ldots, n)T$ is the **i-th** incremental model. The new formal model can be denoted as NF, where $NF \in T$ is a predicate. NF satisfies:

$$NF = Q \wedge C1 \wedge C2 \cdots \wedge Cn \quad (5)$$

NF is refined from Q. So NF satisfies:

$$NF \supseteq Q \quad (6)$$

Each modification in the formal model Q will have a corresponding modification in the UML model P, that is, $for \forall Ci \in T, \exists Di \in S$, there is $Di = F^{-1}(Ci)(i = 1, 2, \ldots, n)$.

Sequence $D1, D2 \ldots, Dn$ is an incremental model Sequence of the UML model P, where each predicate $Di(i = 1, 2, \ldots, n) \in S$ is the **i-th** incremental model. The new UML model after being modified can be denoted as NU, where $NU \in S$. NU satisfies:

$$NU = P \wedge D1 \wedge D2 \cdots \wedge Dn \quad (7)$$

NU is refined from P. So NU satisfies:

$$NU \supseteq P \quad (8)$$

According to Galois connection's property (2), the formal reverse transformation can be abbreviated as follow:

$$D1 \wedge D2 \cdots \wedge Dn = F^{-1}(C1 \wedge C2 \cdots \wedge Cn) \quad (9)$$

Here we use the Galois connection to proof the feasibility and correctness of the reverse transformation process in section 3.

proof of the feasibility:

$$\begin{aligned} F^{-1}(NF) &= F^{-1}(Q \wedge C1 \wedge C2 \cdots \wedge Cn) && \text{by formula(5)} \\ &= F^{-1}(Q) \wedge F^{-1}(C1) \wedge F^{-1}(C2) \\ &\quad \cdots \wedge F^{-1}(Cn) && \text{by formula(2)} \\ &= F^{-1}(Q) \wedge D1 \wedge D2 \\ &\quad \cdots \wedge Dn && \text{because } Di = F^{-1}(Ci) \\ &= P \wedge D1 \wedge D2 \\ &\quad \cdots \wedge Dn && \text{because } P \text{ replace } F^{-1}(Q) \\ &= NU && \text{by formula(7)} \end{aligned}$$

■

$$F^{-1}(NF) = NU \quad (10)$$

Formula (10) denotes that the formal model NF can be indirectly reversely transformed to the UML model NU. It testifies the feasibility of the reverse transformation.

proof of the correctness:

$$\begin{aligned} F(NU) &= F \circ F^{-1}(NF) && \text{by formula(4)} \\ &= NF && \text{because } F^{-1}(NF) = NU \end{aligned}$$

■

$$F(NU) = NF \quad (11)$$

Formula (11) denotes that the UML model NU reversely transformed from the formal model NF can be transformed to the formal model NF again. It testifies the correctness of the indirect reverse transformation.

V. Conclusion and Future Work

There are two advantages of our reverse transformation. Firstly, our reverse transformation is an indirect method. We take full advantage of XMI to avoid the loss of the information, promote the model reuse and speed up the reverse transforming process. Secondly, the result UML model is complete and can be refined by the graphic developer.

To prove the feasibility and correction of our method, we adopt the UTP to be our predicate proof technique. To demonstrate our method, we have developed a prototype, which translates the Z [6, 49, 50] models based on the France's transformation [12] to UML Class Diagrams. For the suitable length of this paper, we don't present it as a case. we will present it in future paper.

In this paper, we don't give the specific rules of the reverse transformation from the differential formal model to the differential UML model. There are different specific rules in different transformations. In the future work, we should establish different specific rules according of different transformations.

References

[1] C. Hoare J. He. *Unifying Theories of Programming*. Printice-Hall International, Europe 1998.

[2] G. Booch J. Rumbaugh and I. Jacobson. *The Unified Modeling Language User Guide*, 2nd ed. Massachusetts:Addsion-wesley, 2005.

[3] M. Nenad S. David R. David F. Robbins and E. Jason.Modeling software architectures in the Unified Modeling Language. In *ACM Transactions on Software Engineering and Methodology*, ACM, New York, 2002, pp. 2-57.

[4] M. Huth M. Ryan *Logic in Computer Science* 2nd ed, Cambridge University Press, England , 2004.

[5] OMG (2009) XMI Metadata Interchange [online]. Available:http://www.omg.org/spc/XMI

[6] J.M. Wing. A Specifier's Introduction to Formal Methods. IEEE, Computer, 1990, pp. 8, 10-23.

[7] M. Hinchey J. Bowen and C. Rouff. *Introduction to Formal Methods*, Springer, London, 2006.

[8] M. Bittner and F. Kammller. Translating Fusion/UML to Object-Z. *First ACM and IEEE International Conference on Formal Methods and Models for Co-Design*, IEEE Press, New York, 2003, pp. 49.

[9] D. Roe. Mapping UML models incorporation OCL Constraints into Object-Z. Technical report, Imperial College, London, 2003.

[10] S. Dupuy Y. Ledru and M. Chabre-Peccoud. An Overview of RoZ: A Tool for Integrating UML and Z Specifications. In *Advanced Information Systems Engineering*, Springer, Heidelberg, 2000, pp. 417–430.

[11] Y. Ledru. Using Jaza to Animate RoZ Specifications of UML Class Diagrams. In *30th Annual IEEE/NASA Software Engineering Workshop SEW-30 (SEW'06)*, Columbia, Maryland, 2006, pp.253-262.

[12] R.B. France E. Grant J.m. Bruel. UMLtranZ: An UML-based rigorous requirements modeling technique. Technical report, Colorado State University, Fort Collins, Colorado, USA, 2000.

[13] A. Evans R.B. France E. Grant. Towards Formal Reasoning with UML Models. In *Proceedings of the OOPSLA'99 Workshop on Behavioral Semantics*, 1999.

[14] J.M. Bruel R.B. France. Transforming UML models to formal specifications. In *UML'98 - Beyond the notation*, Springer,Verlag, 1998.

[15] N. Amalio S. Stepney F. Polack. Modular UML semantics: Interpretations in Z based on template and generics. Technical report, In *Formal Aspects of Component Software 2003*.

[16] N. Amalio F. Polack S. Stepney. U Templates and Generics for Translating UML Class Diagrams into Z. Technical report, Department of Computer Science, University of York, 2003.

[17] K. Lano D. Clark K. Androutsopoulos. UML to B: Formal Verification of Object-Oriented Models. In *Integrated Formal Methods*, Springer, Heidelberg, 2004, pp. 187-206.

[18] V. Dniel M. Asztalos D. Bisztray A. Boronat D. Dang R. Geib. Transformation of UML Models to CSP: A Case Study for Graph Transformation Tools. In *Applications of Graph Transformations with Industrial Relevance*, Springer, Heidelberg 2008, pp. 540-565.

[19] E. Sekerinski R. Zurob. Translating Statecharts to B. In *Integrated Formal Methods*, Springer, Verlag, 2002, pp. 128-144.

[20] L. Hung. Contributions for Modelling UML State-Charts in B. In *Integrated Formal Methods*, Springer, Heidelberg, 2002, pp. 109–127.

[21] D. Bisztray E.Karsten H. Reiko. Case Study: UML to CSP Transformation. Technical Report, University of Leicester, UK, 2007.

[22] S. Kim and C. David. Formalizing the UML Class Diagram Using Object-Z. In *"UML"'99 - The Unified Modeling Language*, Springer, Heidelberg, 1999, pp. 753–753.

[23] S. Kim and C. David. A Formal Mapping between UML Models and Object-Z Specifications. In *ZB 2000: Formal Specification and Development in Z and B*, Springer, Heidelberg, 2000, pp. 2–21.

[24] S. Kim D. Carrington R. Duke. A Metamodel-based transformation between UML and Object-Z. In *IEEE 2001 Symposium on Human Centric Computing Languages and Environments*, Stresa, Italy, 2001, pp.112.

[25] V. Dniel. A Formal Semantics of UML Statecharts by Model Transition Systems. In *Graph Transformation*, Springer, Heidelberg, 2002, pp. 378-392.

[26] VD. Beeck. Formalization of UML-Statecharts. In

"UML" 2001 - The Unified Modeling Language. Modeling Languages, Concepts, and Tools, Springer, Heidelberg, 2001, pp. 406–421.

[27] Y.N. Muan B. Michael. Towards Formalizing UML State Diagrams in CSP. In *First International Conference on Software Engineering and Formal Methods (SEFM'03*, IEEE Press, 2003, pp. 138–138.

[28] S. Harald. Semantics of Control-Flow in UML 2.0 Activities. In *Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing*, IEEE Press, Washington,DC, 2004, pp. 235-242.

[29] J.H. Hausmann S. Harald. Towards a Formal Semantics of UML 2.0 Activities. *Software Engineering 2005* 2005,P-64, 117-128.

[30] S. Harald. Semantics and Verification of Data Flow in UML 2.0 Activities. *Electr.Notes Theor. Comput. Sci.*, 2005, pp. 127(4), 35-52.

[31] S. Harald. Semantics of exceptionin UML 2.0 Activities.Technical Report, Ludwig-Maximilians-University 2004.

[32] S. Harald. Semantics of Structured nodes in UML 2.0 activities. In *2nd Nordic Workshop on UML*, 2004, pp.19-32.

[33] X. Li Z. Liu J. He. A formal semantics of UML sequence diagram. In *Proceedings of the 2004 Australian Software Engineering Conference*, IEEE Press, New York, 2004, pp. 433–442.

[34] C. Eichner H. Fleischhack R. Meyer U. Schrimpf C. Stehno. Compositional Semantics for UML 2.0 Sequence Diagrams Using Petri Nets. In *SDL 2005: Model Driven Systems Design*, Springer, Heidelberg, 2005, pp. 133–148.

[35] R. oscar M.cF. Joao. Some Rules to Transform Sequence Diagrams into Coloured Petri Nets. In *7th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, 2006, pp. 237–256.

[36] O. Flavio. -Method: a model-driven formal method for architecture-centric software engineering. In *-Method: a model-driven formal method for architecture-centric software engineering*, ACM Press, New York, 2006, pp. 1–13.

[37] J. Abrial. Formal methods in industry: achievements, problems, future. In *Proceedings of the 28th international conference on Software engineering*, ACM Press, New York, 2006, pp. 761-768.

[38] J. Dong Y. Li J. Sun J. Sun H. Wang. XML-Based Static Type Checking and Dynamic Visualization for TCOZ. In *Formal Methods and Software Engineering*, Springer, Heidelberg, 2002, pp. 311–322.

[39] H. Fekih L.J.B. Ayed and S. Merz. Transformation of B Specifications into UML Class Diagrams and State Machines. In *Proceedings of the 2006 ACM symposium on Applied computing*, ACM, 2006.

[40] A. Hammad B. Tatibou"et and J.C.Voisinet. From a B Specification to UML StateChart Diagrams. In*Formal Methods and Software Engineering*, Springer,2002.

[41] A. Idani Y. Ledru and D. Bert. Derivation of UML Class Diagrams as Static Views of Formal B Developments. In *Formal Methods and Software Engineering*, Springer, 2005.

[42] A. Idani and Y. Ledru. Dynamic graphical UML views from formal B specifications. In *Information and Software Technology*, Elsevier, 2006, pp. 154–169.

[43] A. Idani and Y. Ledru. Object oriented concepts identification from formal B specifications. In *Formal Methods in System Design*, Springer, 2007, pp. 217–232,

[44] A. Idani and Y. Ledru and D. Bert A Reverse-engineering Approach to Understanding B Specifications with UML diagrams. In *Software Engineering Workshop, 2006. SEW'06. 30th Annual IEEE/NASA*, 2006, pp. 97–106.

[45] OMG (2009) Extensible Markup Language(XML) [online] Available: http://www.w3.org

[46] OMG (2009) Unified Modeling Language(UML) [online] Available: http://www.omg.org/technology

[47] OMG (2009) Meta Object Facility. http:// www.omg.org/cgi-bin

[48] Pagano Dennis and B.Anne. Engineering Document Applications - From UML Models to XML Schemas. In *The Markup Conference*, Balisage, 2009.

[49] J.M. Spivey. *The Z Notation-A Reference Manual*. Prenctic-Hall International, 1998.

[50] J.M. Spivey. *An Introduction to Formal Specification and Z*, 2nd ed. Prentice Hall, NJ, USA, 1996.