Tao JIANG
Yunnan University of Nationalities

Research on Metamodel Consistency Verification Based on First-order Logical Inference

***Abstract**. Domain-Specific Metamodeling Language (DSMML) defined by informal method cannot strictly represent its structural semantics, so its properties such as consistency cannot be holistically and systematically verified. In response, the paper proposes an approach for verifying metamodels consistency based on formalization of DSMML named XMML. Firstly, we describe our approach of formalization, and then, the approach of consistency verification of XMML and its metamodels based on first-order logical inference is presented, finally, the formalization automatic mapping engine for metamodels is designed and relevant test is performed to show the feasibility of our formal method.*

***Streszczenie.** W artykule zaproponowano metodę weryfikacji metamodelu DSMML (Domain Specific Metamodeling Language). Weryfikacja bazuje na interferencji logicznej pierwszego rzędu (**Badania weryfikacji konsystencji metamodelu bazujące na interferencji logicznej pierwszego rzędu**)*

Keywords: Domain-Specific Metamodeling Language (DSMML); meta-type; consistency verification; automatic mapping.
Słowa kluczowe: metamodel, DSMML.

1 Introduction

As a metamodeling language for DSM [1] and a metalanguage used to build Domain-Specific Modeling Languages (DSMLs), DSMML plays an important role in software system modeling of specific domains.

Semantics of DSMML can be divided into structural semantics [2] and behavioral semantics. The former describes static semantic constraints between metamodeling elements, focusing on the static structural properties; the latter concerns execution semantics of domain metamodels, focusing on the dynamic behavior of the metamodels. Although structural semantics is very important, research in structural semantics is not so extensive and deep as behavioral semantics', so this paper only studies structural semantics of DSMML.

There are several problems that have not been solved well for DSMML, which include precise formal representation of its semantics, approach of properties analysis and verification of metamodels based on formalization, and automatic mapping of formalization of metamodels.

The paper proposes an approach for verifying metamodels consistency based on first-order logical inference, based on this, design of formalization automatic mapping engine for metamodels and its test are introduced.

2 Related Works

Within the domain-specific language community, graph-theoretic formalisms have received the most research attention [3]. The majority of work focuses on model transformations based on graph, but analysis and verification of properties of models has not received the same attention. For example, Z [4] or B [5] formalizations of UML could be a vehicle for studying rich syntax, but automated analysis and verification is less likely to be found.

There are much typical work on formalization of modeling language, such as Andre's formalization and verification of UML class diagram based on ADT [6], Malcolm Shroff's formalization and verification of UML class diagram based on Z [7], Paige's formalization of BON based on PVS [8] and Jackson.E.K's formalization of DSML based on Horn logic [9] and so on. Without considering formalization of metamodeling language and automatic translation from metamodels to the corresponding formal semantic domain, these approaches have lower level of automated analysis and verification.

3 Formalization of XMML Based on First-order Logic

We introduce abstract syntax of XMML and give a formal definition of XMML, based on this, the approach for formalizing XMML is presented.

3.1 Abstract Syntax of XMML

XMML is divided into four layers: metamodeling language layer used to define different DSMLs where XMML is located, DSML layer used to build concrete domain application models, domain application model layer used to make corresponding source codes of target system by code generator, and target application system layer [10]. Layered Architecture of XMML is shown in Fig 1.

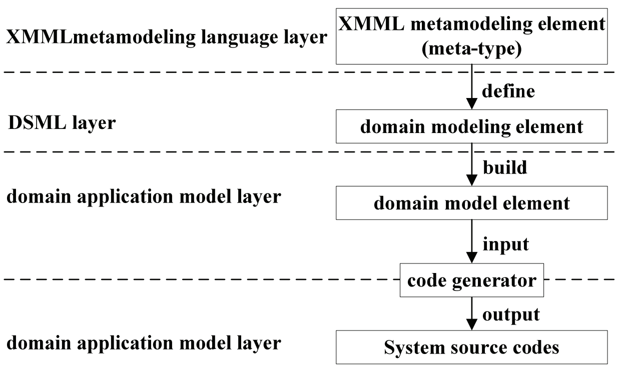
XMMLmetamodeling language layer
XMML metamodeling element (meta-type)
define
DSML layer
domain modeling element
build
domain application model layer
domain model element
input
code generator
output
domain application model layer
System source codes

Fig.1. Layered architecture of XMML

We require that element of XMML is called metamodeling element and element of DSML built based on XMML is called domain modeling element and domain object built based on DSML is called domain model element. Among them, metamodeling element is also called meta-type and type of model element is modeling element and type of modeling element is meta-type [10].

Metamodeling element of XMML is divided into two types: entity type and association type, the former is used to describe modeling entities in metamodel and the latter concerns relationships between modeling entities. Metamodeling element of entity type consists of four types such as model type, entity type, reference entity type and relationship type. Metamodeling element of association type includes the following five types: role assignment association, model containment relationship, attachment relationship, reference relationship, and refinement relationship. We finish formalization of structural semantics of XMML based on the above nine meta-types.

3.2 A Formal Definition of XMML

XMML can be regarded as composition of the following five parts: a set of predicate symbols S_{XMML} denoting corresponding metamodeling elements, an extended set of predicate symbols S_{XMML}^c used to derive properties, a set of closed first-order logic formulas F_{XMML} denoting constraints over all metamodels, a set of constants O_{XMML} denoting public properties, a set of terms symbols Ω_{XMML} denoting modeling elements used to build metamodel. Among them, S_{XMML}^c and O_{XMML} can be empty, F_{XMML} is defined by first-order logic implication formulas based on S_{XMML}, S_{XMML}^c and O_{XMML}. The definition focuses on formal characterization of structural properties of XMML.

Definition 1(**XMML**). XMML L_{XMML} is a 5-tuple of the form $<S_{XMML}, S_{XMML}^c, \Omega_{XMML}, O_{XMML}, F_{XMML}>$ consisting of S_{XMML}, S_{XMML}^c, O_{XMML}, F_{XMML} and Ω_{XMML}.

S_{XMML} and S_{XMML}^c as a group of predicate symbols, O_{XMML} as a group of constant symbols, and F_{XMML} as a group of constraint axioms are all added to first-order logic formalized system called predicate calculus Q [11][12] to form formalized system of XMML called T_{XMML} based on predicate calculus Q. Once S_{XMML}, S_{XMML}^c, O_{XMML} and F_{XMML} are derived, we finish formalization of L_{XMML} based on first-order logic. In addition, set of all metamodels of XMML can be expressed as powerset of the term algebra $M_{XMML}=\mathcal{P}(T_S(\sum_{XMML}))$ as a group of interpretations of T_{XMML}, so we can determine whether any metamodel $m_{XMML} \in M_{XMML}$ is well-formed instance of XMML..

3.3 Formalization of Meta-types of XMML

For four meta-types of entity type, each type of them can be represented as a unary predicate. Therefore, for each *Model*, a unary predicate *Model*(x) is defined to denote meta-type of modeling element x is *Model*, i.e. *Model*(x) $\in S_{XMML}$, which can contain all other modeling elements of entity type. Similarly, we can derive *RefEntity*(x) $\in S_{XMML}$ and *Entity*(x) $\in S_{XMML}$ and *Relationship*(x) $\in S_{XMML}$.

It is much more complex for five meta-types of association type and their constraint relationships to formalize; each type of them has to be expressed using a group of first-order logic implication formulas, in addition, semantic implication relationships and consistency between formulas have to be discussed. For example, for each reference relationship (denoted *Reference*) from modeling element of reference entity type x to entity type y, a binary predicate *Reference*(x, y) is defined to represent that element x points to element y by *Reference* edge, i.e. *Reference*(x, y) $\in S_{XMML}$.

By formalizing all meta-type of association type in the same way, we can establish formula subset of role assignment association named *RoleRelaSet*, formula subset of model containment named *ContainmentSet*, formula subset of attachment named *AttachmentSet* and formula subset of refinement constraints named *RefinementSet* one by one. Thus, set of constraint axioms of T_{XMML} named F_{XMML} can be considered as union of all of the above subsets, i.e.

F_{XMML} = *ContainmentSet* $\cup$ *ReferenceSet* $\cup$ *AttachmentSet* $\cup$ *RoleRelaSet* $\cup$ *RefinementSet.*

Formalized system of XMML called T_{XMML} based on predicate calculus Q is established after formalization of all meta-type of XMML. According to predicate calculus Q [12] and composition of XMML, T_{XMML} is defined as follows.

Definition 2(***T*$_{XMML}$**). T_{XMML} consists of formal language L and deduction structure defined based on L. Except some modification of symbols set $S_{XMML}(L)$ and adding of a set of constraints axioms, T_{XMML} is based entirely on predicate calculus Q. Its symbols set $S_{XMML}(L)$ is union of logical link symbol set $L_{XMML}(L)$ and collection of individual variables $V_{XMML}(L)$ and Individual constant symbol set $C_{XMML}(L)$ and predicate symbols set $P_{XMML}(L)$, i.e. $S_{XMML}(L) = L_{XMML}(L) \cup V_{XMML}(L) \cup C_{XMML}(L) \cup P_{XMML}(L)$, among them, $L_{XMML}(L)$ and $V_{XMML}(L)$ are same as Q's, $C_{XMML}(L) = O_{XMML}$, $P_{XMML}(L) = S_{XMML} \cup S_{XMML}^c$. Axioms set A_{XMML} of T_{XMML} is composed of logical axioms set LA_{XMML} of Q and constraint axioms set F_{XMML} of XMML, i.e. $A_{XMML} = LA_{XMML} \cup F_{XMML}$.

The semantic interpretation of T_{XMML} is a metamodel built based on XMML, universe of discourse of interpretation is the set of all entity modeling elements contained in the metamodel. Similarly, metamodel built based on XMML can be formalized via metamodel mapping from metamodel to a set of predicate statements. Once XMML and metamodel are formalized based on first-order logic, we can implement logical consistency verification of XMML and metamodels of XMML based on first-order logical inference.

4 Consistency Verification of XMML and Metamodels

4.1 Consistency Verification of XMML

It is not easy to find a true interpretation for constraint axiom set F_{XMML} of T_{XMML} to prove semantic consistency of T_{XMML} , on the other hand, It is very difficult to derive grammatical consistency of F_{XMML} by hand-proving due to too many formulas contained in F_{XMML}, so we can only prove logical consistency of T_{XMML} based on automatic theorem prover. Reference to the literature [13], we give the following definition.

Definition 3 (**logical consistency of XMML**). XMML is logically consistent iff the constraint axiom set F_{XMML} of T_{XMML} is proved to be logically consistent in the automatic theorem prover; XMML is logically inconsistent iff the constraint axiom set F_{XMML} of T_{XMML} is proved to be contradictory in the automatic theorem prover, denoted $F_{XMML} \vdash$ *False*.

If T_{XMML} is proved to logically consistent, then XMML must have an interpretation that can be satisfied, thus it is meaningful to discuss properties of metamodels built based on XMML.

4.2 Consistency Verification of Metamodels

It is obvious that we can only determine whether a metamodel is a legitimate instance of XMML by examining relationship that the metamodel satisfies XMML. From the perspective of formalization, a metamodel is an interpretation of T_{XMML}, and a legal metamodel is an interpretation that satisfies all constraint formulas of F_{XMML} of T_{XMML}, so the relationship that metamodel satisfies XMML is equivalent to the relationship that the interpretation of T_{XMML} satisfies T_{XMML}. According to semantic theory of first-order logic [12], we give a formal definition of interpretation of T_{XMML}.

Definition 4 (**interpretation of *T*$_{XMML}$**). An interpretation of T_{XMML} is a 2-tuple of the form < E, I >. Among them, E is universe of discourse of interpretation denoting a non-empty set composed of all modeling elements of entity type and constants in the metamodel, I is a mapping that is satisfying the following conditions and defined on set of non-logical symbols NL_{XMML}(L) of T_{XMML}.

1) For any constant c in $C_{XMML}(L)$, i.e. $\forall c \in C_{XMML}(L)$, $I(c)= c^I$ and c^I is an element in E, i.e. $I(c) \in E$;
2) For any unary predicate P in $P_{XMML}(L)$, i.e. $\forall P(x) \in P_{XMML}(L)$, $I(P(x)) = P^I$ and P^I is a subset of E, i.e. $P^I \subseteq E$;

3) For any binary predicate R in $P_{XMML}(L)$, i.e. $\forall R(x,y) \in P_{XMML}(L)$, $I(R(x,y)) = R^I$ and R^I is a binary relation on E, i.e. $R^I \subseteq E \times E$;
4) For any ternary predicate S in $P_{XMML}(L)$, i.e. $\forall S(x,y,z) \in P_{XMML}(L)$, $I(S(x,y,z)) = S^I$ and S^I is a ternary relation on E, i.e. $S^I \subseteq E \times E \times E$;

By Definition 4, an interpretation of T_{XMML} can be considered as a classification of modeling element set constituting a metamodel based on non-logical symbol set of T_{XMML}. Thus, interpretation of every unary predicate can be regarded as set of modeling elements of entity type belonging to some meta-type in the metamodel, and interpretation of every binary predicate is set of some binary relationship between modeling elements in the metamodel, and interpretation of every ternary predicate is set of path relationship formed between modeling elements in the metamodel. According to definition in predicate calculus Q [12], we give a formal definition of relationship that the interpretation of T_{XMML} satisfies T_{XMML} as follows.

Definition 5 (relationship that the interpretation of T_{XMML} satisfies T_{XMML}). Assume that T_I is an interpretation of T_{XMML}, if T_I makes any formula φ in constraint axiom set F_{XMML} of T_{XMML} true under universe of discourse E, that is, for $\forall \varphi \in F_{XMML}$, $T_I \models \varphi$, then the interpretation T_I satisfies T_{XMML}, denoted $T_I \models T_{XMML}$; instead, if T_I makes one formula φ in constraint axiom set F_{XMML} of T_{XMML} under universe of discourse E false, that is, for $\exists \varphi \in F_{XMML}$, $T_I \not\models \varphi$, then the interpretation T_I does not satisfy T_{XMML}, denoted $T_I \not\models T_{XMML}$. We can determine whether T_I satisfies any formula φ in F_{XMML} by the following rules.

1) If φ is unary atomic formula $P(x)$, then $T_I \models \varphi$ iff $x^I \in P^I$, $T_I \not\models \varphi$ iff $x^I \notin P^I$;
2) If φ is binary atomic formula $R(x, y)$, then $T_I \models \varphi$ iff $< x^I, y^I > \in R^I$, $T_I \not\models \varphi$ iff $< x^I, y^I > \notin R^I$;
3) If φ is ternary atomic formula $S(x, y, z)$, then $T_I \models \varphi$ iff $< x^I, y^I, z^I > \in S^I$, $T_I \not\models \varphi$ iff $< x^I, y^I, z^I > \notin S^I$;
4) If $\varphi = \neg \psi$, then $T_I \models \varphi$ iff $T_I \not\models \psi$, $T_I \not\models \varphi$ iff $T_I \models \psi$;
5) If $\varphi = \psi \rightarrow \eta$, then $T_I \models \varphi$ iff $T_I \not\models \psi$ or $T_I \models \eta$, $T_I \not\models \varphi$ iff $T_I \models \psi$ and $T_I \not\models \eta$;
6) If $\varphi = \psi \vee \eta$, then $T_I \models \varphi$ iff $T_I \models \psi$ or $T_I \models \eta$, $T_I \not\models \varphi$ iff $T_I \not\models \psi$ and $T_I \not\models \eta$;
7) If $\varphi = \psi \wedge \eta$, then $T_I \models \varphi$ iff $T_I \models \psi$ and $T_I \models \eta$, $T_I \not\models \varphi$ iff $T_I \not\models \psi$ or $T_I \not\models \eta$;
8) If $\varphi = \forall x \psi$, then $T_I \models \varphi$ iff for all assignments of x, $T_I \models \psi$, $T_I \not\models \varphi$ iff there is an assignment of x such that $T_I \not\models \psi$;
9) If $\varphi = \exists x \psi$, then $T_I \models \varphi$ iff for one assignments of x, $T_I \models \psi$, $T_I \not\models \varphi$ iff for all assignments of x, $T_I \not\models \psi$.

The above determine rules are the basis for establishing metamodels automatic mapping mechanism. Here, we illustrate composition of interpretation and determination of satisfaction relationship by an example. As a small subset of T_{XMML}, T_{SubSet} contains two meta-types of entity type composed of *Model* and *Entity* and two meta-types of association type composed of *Containment* and *Attachment*. Symbol set and constraint axioms set of T_{SubSet} are defined as follows.

- Individual constant symbol set $C_{SubSet}(L) = \varnothing$; predicate symbols set $P_{SubSet}(L)$ = {*Model*(x), *Entity*(x), *Containment*(x,y), *Attachment* (x,y)}
- Constraint axioms set F_{SubSet} ={

1) $\forall x.Model(x) \vee Entity(x)$ (TS$_1$)
completeness of classification
2) $\forall x.Model(x) \rightarrow \neg Entity(x)$ (TS$_2$)
uniqueness of classification
3) $\forall x, y.Containment(x,y) \rightarrow Entity(x) \wedge Model(y)$ (TS$_3$)
containment type constraint
4) $\forall x, y.Attachment(x,y) \rightarrow Entity(x) \wedge Entity(y)$ (TS$_4$)
attachment type constraint
5) $\forall x.\neg Attachment(x,x)$ (TS$_5$) self-attached }

A metamodel built based on T_{SubSet} is shown in Fig 2, which can be formalized as an interpretation of T_{SubSet} named S_I consisting of the following several sets.

- Universe of discourse of interpretation E= {*SoftwareArchitecture, Component*};
- Interpretation unary predicate in $P_{SubSet}(L)$: *Model*^{I} ={*SoftwareArchitecture*}, *Entity*^{I} = {*Component*};
- Interpretation binary predicate in $P_{SubSet}(L)$: *Containment*^{I} = {<*Component*, *SoftwareArchitecture*>}, *Attachment*^{I}= {<*Component*, *Component* >}

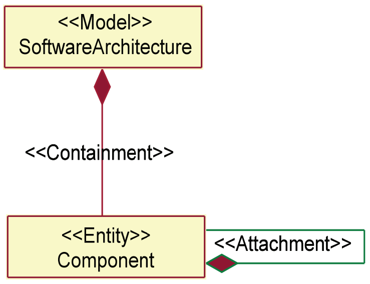

Fig.2. A metamodel of T_{SubSet}

Now we determine whether the metamodel in Fig 2 satisfies T_{SubSet} by Definition 5. Two elements in E belong to *Model*^{I} or *Entity*^{I}, i.e. *SoftwareArchitecture* $\in$ *Model*^{I} and *Component* $\in$ *Entity*^{I}, by determination rule 1, 6 and 8, S_I satisfies TS$_1$, denoted $S_I \models$ TS$_1$; similarly, we can derive $S_I \models$TS$_2$, $S_I \models$TS$_3$, $S_I \models$TS$_4$ and $S_I \not\models$TS$_5$. By Definition 5, we can see S_I does not satisfy T_{SubSet}, denoted $S_I \not\models T_{SubSet}$, so a metamodel in Fig 2 is not a legal instance of T_{SubSet}. Based on the above discuss about relationship that metamodels satisfy XMML, we establish the concept of consistency of metamodel.

Definition 6 (logical consistency of metamodel). If a metamodel satisfies XMML, from the formal point of view, that is, the interpretation of T_{XMML} named T_I satisfies T_{XMML}, denoted $T_I \models T_{XMML}$, then the metamodel is logical consistent; instead, if a metamodel does not satisfy XMML, that is, the interpretation of T_{XMML} named T_I does not satisfy T_{XMML}, denoted $T_I \not\models T_{XMML}$, then the metamodel is logical inconsistent.

Assume that T_I is an interpretation of T_{XMML} and $T_L(M)$ is a group of first-order predicate statements that T_I corresponds to and all interpretation subsets in T_I are not empty, then the mapping δ_E from T_I to $T_L(M)$ is a bijection based on universe of discourse E.

1) For every element in interpretation subset P^I corresponding to one unary predicate P in T_I , i.e. $\forall a \in P^I$, by δ_E it is mapped to an unary predicate statement combined with P and a in $T_L(M)$, i.e. $\delta_E(a) = P(a)$;
2) For every element in interpretation subset R^I corresponding to one unary predicate R in T_I , i.e. $\forall < a,b > \in R^I$, by δ_E it is mapped to a binary predicate statement combined with R and <a,b> in $T_L(M)$, i.e. $\delta_E(< a,b >) = R(a,b)$;
3) For every element in interpretation subset S^I corresponding to one ternary predicate S in T_I , i.e. $\forall < a,b,c > \in S^I$, by δ_E it is mapped to a ternary

predicate statement combined with S and<a,b,c> in $T_L(M)$, i.e. $\delta_E(<a,b,c>)=S(a,b,c)$;

4) For every formula f_i in the additional constraint formula set F_{Atta} of T_l, i.e. $\forall f_i \in F_{atta}$, by δ_E it is mapped to the same formula, i.e. $\delta_E(f_i)=f_i$.

Similarly, the inverse mapping δ_E^{-1} of δ_E from $T_L(M)$ to T_l is also a bijection based on universe of discourse E. T_l and $T_L(M)$ are completely equivalent but different representation derived from same formal result, so the relationship that T_l of T_{XMML} satisfies F_{XMML} and logical consistency of union of F_{XMML} and $T_L(M)$ that T_l are completely equivalent in first-order logic. Its proof of equivalence is presented as follows.

Theorem 1 (**equivalence of satisfaction and logical consistency**). Union of axiom set F_{XMML} of T_{XMML} and predicate statements set $T_L(M)$ that T_l corresponds to are logical consistent iff T_l satisfies axiom set F_{XMML} of T_{XMML}.
Proof. (Sufficiency) $T_l \models T_{XMML}$, by Definition 6, T_l makes any formula φ in constraint axiom set F_{XMML} of T_{XMML} true under universe of discourse E, that is, for $\forall\varphi\in F_{XMML}$, $T_l \models \varphi$. If φ is a non-implication formula, $T_l \models \varphi$ means that subset of $T_L(M)$ that T_l corresponds to is a replacement of φ based on E which makes a true value form of φ; if φ is an implication formula $\psi \rightarrow \eta$, $T_l \models \varphi$ means that T_l makes premise ψ of φ false or makes premise ψ and conclusion η true of φ, for the latter, subset of $T_L(M)$ is also a replacement of φ based on E which similarly makes a true value form of $\psi \rightarrow \eta$. True value form of $T_L(M)$ generated via any formula φ and formula φ itself must be consistent, so Union of F_{XMML} and $T_L(M)$ that T_l corresponds to are logical consistent.
Proof. (necessity) Union of F_{XMML} and $T_L(M)$ that T_l corresponds to are logical consistent, thus, union of $T_L(M)$ and any formula φ in F_{XMML} are logical consistent, so true value form of $T_L(M)$ generated via any formula φ can be considered as a replacement of φ based on E, T_l must make φ true under universe of discourse E, that is, for $\forall\varphi\in F_{XMML}$, $T_l \models \varphi$, by Definition 6, we can derive $T_l \models T_{XMML}$.

Similarly, we can also prove equivalence of non-satisfaction and logical inconsistency, so method of consistency verification of metamodel can be derived as follows.

Inference 1 (**logical consistency of metamodel**). If union of constraint axiom set F_{XMML} of T_{XMML} and set of first-order predicate statements $T_L(M)$ generated via metamodel M is logically consistent, then the metamodel M is consistent; instead, if union of constraint axiom set F_{XMML} of T_{XMML} and set of first-order predicate statements $T_L(M)$ generated via metamodel M is logically inconsistent, denoted $F_{XMML} \cup T_L(M) \vdash False$, then the metamodel M is inconsistent.

5 Design and Implementation of *MapM*

Formalization automatic mapping tool for metamodel called *MapM* (*Mapping of Metamodels*) is designed and implemented to finish automatic translation from metamodel based on *XMML* concrete syntax scheme to the corresponding set of first-order predicate statements $T_L(M)$ in SPASS format [14]. SPASS is an efficient open source automatic theorem prover developed by MPI-INF—a team of teachers at Saarland University [15]. Thus we can realize automatic process of analysis and verification of consistency of metamodel built based on XMML. Logical architecture of *MapM* is shown in Fig 3.

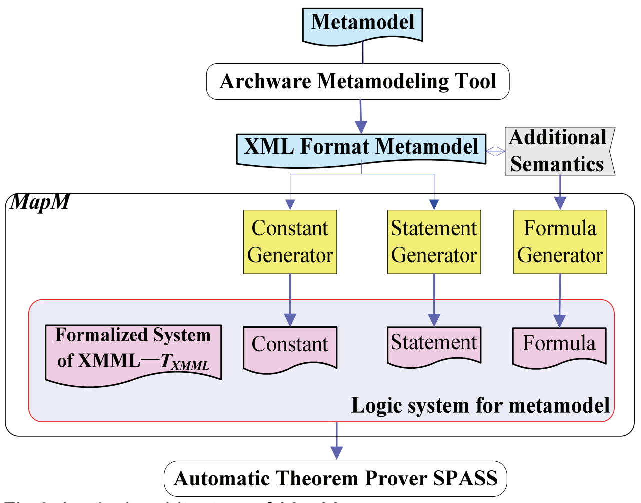

Fig.3. Logical architecture of *MapM*

Based on .net 2.0 platform, by using C#.net as development language, we implement the corresponding prototype system for *MapM* and integrate them in the modeling environment named *Archware* [10] of XMML, thus it becomes possible for *Archware* to verify metamodels built based on XMML.

6 Test and Analysis

The total number of first-order logic formulas generated via metamodel based on formalization mapping rules has a square relation with number of entity modeling elements [15]; on the other hand, there lies in an exponential relationship between the time it takes for completing proof and the number of logic formulas, i.e. $T(M)=O(s^{|F(M)|})$, among them, $F(M)$ is the number of formulas generated via metamodel, $T(M)$ is the time required for validating metamodel M, and s is a constant greater than 1. It can clearly be seen that growth in the number of elements contained in metamodel will lead to growth in square relation of number of formulas that will result in exponential growth of the time required for verification using SPASS.

We find by test that it takes a relatively short time for SPASS to finish logical proof of metamodel of no more than 10 elements. If the metamodel contains contradictions, SPASS can terminate running within a few seconds and output *proof found*; otherwise, if no contradictions are contained in it, it takes a relatively long time for SPASS to finish proof and output *completion found*. If more than 10 elements are contained in metamodel and there exist contradictions among them, it often takes 10-60 seconds for SPASS to finish proof, however, if there are no contradictions in the metamodel, SPASS have to spend much longer time.

We also perform consistency verification of the same metamodel by translating metamodel to set of first-order predicate statements in Otter format based on another automatic theorem prover named Otter, which is a automated deduction software developed by professor William McCune at University of New Mexico [16]. We find that it takes a little bit longer time for Otter to finish logical proof of metamodel compared with SPASS when no more than 10 elements are contained in the metamodel. On the other hand, if more than 10 elements are contained in the metamodel and there exist contradictions among them, Otter spends much longer time on verification than SPASS. Diagram of relationship between prover proof time and number of entity elements in the metamodel containing contradictions is shown in Fig 4. From Fig 4, we can find that growth of prover proof time is very slow when number of entity elements in metamodel is less than 10 and time increases rapidly with growth of number once number of elements exceeds 10. We can also see that SPASS is more efficient than Otter in automatic proof, so we regard SPASS as our prover.

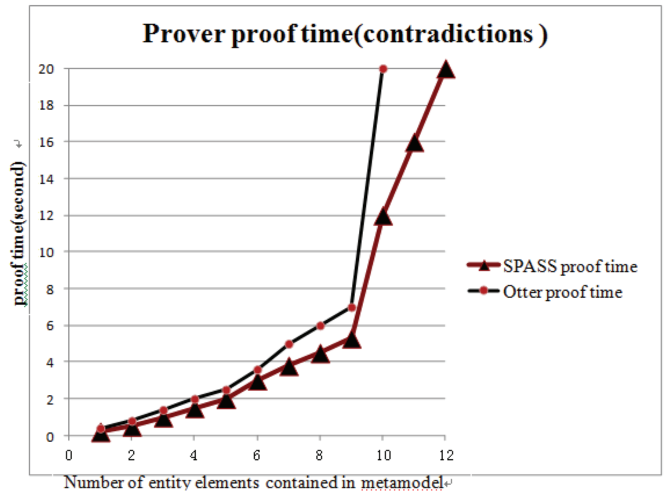

Fig.4.Relationship between prover proof time and elements number

Professor Zhu also performs property verification of UML metamodel and model using SPASS as his prover [13]. Similar to ours, his test result shows that the number 10 is the critical value of number of elements contained in metamodel.

So we conclude that the size of the metamodel which can be verified based on SPASS is limited, that is to say, the number of elements contained in metamodel is best not more than 10, otherwise, much longer time have to be spent or even running can never be terminated unless timeout is set.

7 Conclusions

DSMML defined in the informal way cannot precisely describe its structural semantics, which makes it difficult to holistically and systematically verify its properties such as consistency. In response, the paper proposes an approach for verifying metamodels consistency based on formalization of DSMML named XMML, and we illustrate our approach by consistency verification of a metamodel instance based on first-order logic inference; based on this, design of corresponding formalization automatic mapping engine for metamodels and relevant test are introduced to show the feasibility of our formal method.

ACKNOWLEDGMENT

The author would like to thank Prof. Hua Zhou, Dr. Xinping Sun and Dr. Yong Yu for valuable discussions. This work was supported by Yunnan Provincial Department of Education Research Fund Key Project (*No. 2011z025*) *and General Project* (*No. 2011y214*).

REFERENCES

[1] dsmforum, Enterprise apps in smartphones, http://www.dsmforum.org/phone.html..
[2] Jackson.E.K, Sztipanovits.J, "Formalizing the Structural Semantics of Domain-Specific Modeling Languages", Journal of Software and Systems Modeling, 2008.
[3] B´E ZIVIN, J., AND GERB´E, O, "Towards a precise definition of the omg/mda framework", in Proceedings of the 16th Conference on Automated Software Engineering (ASE 01) (2001), pp. 273–280.
[4] EVANS, A., FRANCE, R. B., AND GRANT, E. S, "Towards formal reasoning with uml models", in Proceedings of the Eighth OOPSLA Workshop on Behavioral Semantics.
[5] MARCANO, R., AND LEVY, N, "Using b formal specifications for analysis and verification of uml/ocl models", in Workshop on consistency problems in UML-based software development.5th International Conference on the Unified Modeling Language (2002), pp. 91–105.
[6] W.Andreopoulos, "Defining Formal Semantics for the Unified Modeling Language", in Technique Report of University of Toronto, 2000, Toronto.
[7] Malcolm Shroff, "Towards A Formalization of UML Class Structures in Z", in Proceedings of COMPSAC'97.
[8] R. F. Paige, B. P. J, "Metamodel-Based Model Conformance and Multiview Consistency Checking", ACM Transactions on Software Engineering and Methodology, 2007. 16(3): p. 1-49.
[9] Jackson.E.K, Sztipanovits.J, "Towards a formal foundation for domain specific modeling languages", Proceedings of the Sixth ACM International Conference on Embedded Software (EMSOFT'06) (October 2006) 53-62.
[10] Sun XP, A Research of Visual Domain-Specific Meta-Modeling Language and Its Instantiation, Kunming: Yunnan University.2008.
[11] Gu TL, Formal methods of software development, Higher Education Press, Beijing, 2005.
[12] Cheng MZ, Yu JW, Logic foundation—first-order logic and first-order theory, Chinese People University Press, Beijing, 2003.
[13] H. Zhu, L. Shan, I. Bayley, and R. Amphlett, "A Formal Descriptive Semantics of UML and Its Applications", in UML 2 Semantics and Applications, K. Lano (Eds). 2008, John Wiley & Sons, Inc.
[14] Christoph Weidenbach, SPASS: Tutorial, 2000.
[15] Spass. 2008, http://www.spass-prover.org.
[16] W.McCune. OTTER 3.0 reference manual and guide. Argonne National Laboratory Tech. Argonne,IL,USA,1994, http://www.cs.unm.edu/~mccune/otter.

***Authors**: Ph.D. Tao Jiang, School of Mathematics and Computer Science, Yunnan University of Nationalities, 134 Yi Er Yi Avenue, Kunming, P.R.China, 650031, E-mail: jtzwy123@gmail.com.*