Engineering Methods For Schema Transformation: Application to XML

Naveen Prakash[1] and Sangeeta Srivastava[2]
[1] *JIIT, A10, Sector 62, NOIDA 201307, India*
praknav@hotmail.com
[2] *BCAS, Veer Sarvarkar Complex, Pusa, New Delhi, India*
sangeetasrivastava@hotmail.com

Abstract: Rather than produce schema transformation methods for every different pair of models, it is proposed that a generic method should be developed using a method engineer friendly model called the Method View Model. Method concepts are organised in an *is composed of* hierarchy. A technique is presented that maps concepts of one hierarchy to those of the other. This technique produces feasible mappings, which can be presented to the method engineer for subsequent fine-tuning. The technique is illustrated for mapping between ER and Relational as well as between Relational and XML models respectively.

Key words: Method Engineering, Schema Transformation, XML

1. INTRODUCTION

Schema transformation is done in a number of situations (i) in moving from one stage in the system life cycle to the next, (ii) building a common global schema for several schemata in heterogeneous data bases, (ii) reverse engineering. With the emergence of e-systems, schema transformation is used when sharing data in B2B systems and

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35614-3_21
C. Rolland et al. (eds.), *Engineering Information Systems in the Internet Context*

in transforming structured schemata to semi-structured HTML/XML schemata. There are three ways in which transformation products are used, and we refer to these as complementary, isolated, and integrated respectively. These are as follows:

- **Complementary**: both the original and its transformed schema are used. This happens when transforming structured schemata to semi-structured ones (Wil01).
- **Isolated**: the transformed schema becomes the focus of attention, for example, when moving from the ER to the relational schema (Korth91). The relational schema with no further reference to the ER Schema is used to build a relational database.
- **Integrated**: the transformed schemata are integrated together in a common data model, as happens in building a global schema for schemata of distributed heterogeneous databases (Mcb 01).

The method for transformation is to start with the pair of models in which the schemata are to be expressed and

(a) Establish mapping between their concepts

(b) Define transformation rules between mapping concepts

(c) Perform the transformation process by applying the rules to a given schema.

The application engineer performs (c). The mapping and transformation rules are given to the application engineer. Step (a) and (b) imply that for every different pair of models, different concept mapping and transformation rules are devised. Indeed a number of proposals exist that convert ER to the Relational model (Kor91), Relational to XML (Mcb01), OO to XML (Ren01). These attempts exploit the experience of their designers and are based on an ad-hoc approach. Evidently, a generic way of doing this will be helpful that can handle any pair of models so that the prevalent experience based, ad-hoc approach gives way to a systematic, computer-supported one.

In (Pra97), Prakash has defined a method as a triplet, <D, Dep, E> where D is a set of decisions, Dep is a set of dependencies between these, and E is an enactment algorithm. Methods were considered as atomic or compound. The latter can be decomposed into simpler methods and are characterised by having more than one product model in them. There are two kinds of compound methods, constructional and transformational. ***Constructional methods*** provide different views of the same system. An example is OMT, which provides three complementary views of an information system from the perspectives

of its Object, Functional and Dynamic Models. These views are related to one another. For example, the objects of the Object Model are expressed as data stores/data flows in the Functional Model and vice-versa. This makes for **mapping** in the sense of (a) above, between the Object and Functional models and is the case of **complementary** transformation. ***Transformational methods*** convert schemata of one model to another as is done, for example, in moving from ER schemata to their relational schemata. This is the case of **isolated** transformation. It can be seen that compound methods whether constructional or transformational are essentially concerned with schema transformation in its different forms.

Now, the discipline of method engineering enables method engineers to use CAME tools for building methods. By and large, method engineering has been successful in building atomic methods. However, it is possible to construct only a part of **constructional compound methods**: in OMT, for example, methods for the three separate component product models are engineered but their ***mappings*** are not. Similarly, method-engineering techniques for building **transformational compound methods** have not yet been reported.

Our research is directed at engineering transformational compound methods. Thus, we aim to address issues (a) to (c) mentioned above. However, in this paper we look at (a) only. That is, given a pair of models, we show that it is possible to provide CAME support to establish mapping between their concepts. We extend the approach of Gupta and Prakash (01) to do this. They defined a Method View Model that provides the concepts in terms of which method engineers visualise their methods. This Model defines two relationships, *is composed of* and *is mapped to* between method concepts. The former is used for building atomic methods and has been explored in (Gup01). We will use the latter as a means of establishing a **mapping** between concepts of compound methods. We show in this paper, that the *is composed of* can be used to derive the *is mapped to*. Indeed, it is possible to automate this derivation so that a CAME tool can produce a feasible mapping between concepts of a pair of methods. The input to such a tool is the *is composed of* hierarchies of the two methods.

The feasible **mapping** is presented to the method engineer for review who can modify it to yield the preferred **mapping**.

In the next section we provide an overview of the decision-oriented view of a method. The method view model and the *is composed of* hierarchy is presented in Section 3. Establishment of mapping between concepts of a pair of models is dealt in Section 4. In Section 5, we show that the generic approach of Section 4 can be successfully applied to convert a Relational schema to an XML schema.

2. OVERVIEW OF METHODS

As shown in Figure 1, it is possible to look upon a method in two ways, as *atomic/compound* or as *constructional/transformational*. A *compound* method consists of more than one component method whereas an atomic method cannot be decomposed further. An *atomic* method deals only with those products that are expressed in exactly one product model. A *transformational* method is used for transforming a product, expressed in one or more product models, into a product of other product model(s). Such methods build isolated products. In contrast, a *constructional* method is used whenever a new product, expressed in one or more product models, is to be constructed. These methods build complementary products. A *constructional* method that builds products for the ER model is *atomic* since the product is expressed in exactly one model. Similarly, the *transformational* method for converting an ER product into a Relational product is *atomic* since each of the products is expressed in exactly one product model. An OMT product is compound constructional since the product is expressed in more than one product model.

A method M is defined as (Pra97)

M= Op (M1, M2...Mn)

Where Op can be Construct or Transform. For a constructional method M

M= Construct (M1, M2...Mn).

It should be possible to map every pair Mi, Mj (i<>j) in M to one another.

That is,
Map (Mi, Mj)
Similarly, for a transformational method
M= Transform (M1, M2...Mn)
It should be possible to convert from every Mi to Mi+1 in M
Convert (Mi, Mi+1).
Constructional and *Transformational* methods impose constraints on Map and Convert respectively for them to be well defined.

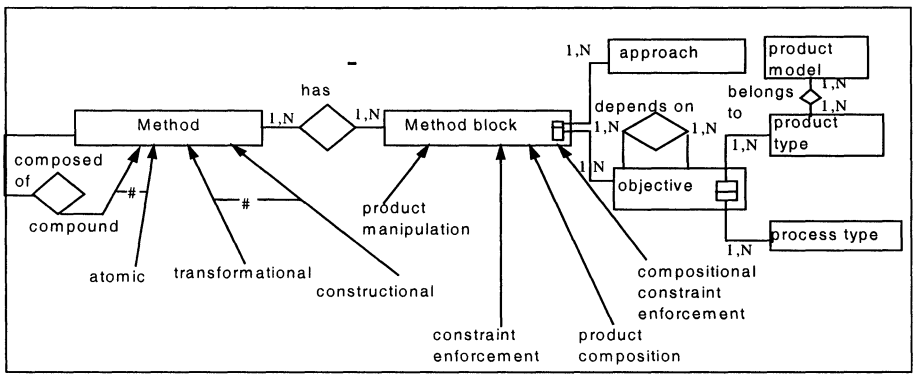


Figure 1:The Generic Method

Compound Methods are well-defined iff component methods M1, M2...Mn and the Map/Convert are well defined. Whereas (Gup01) has looked after well definedness of M1, M2...Mn they left the issue of Map/Convert unaddressed. Map/Convert is the formal way of expressing steps (a) and (b) mentioned in Section 1.

3. THE METHOD VIEW MODEL (MVM)

In (Gup01) a Method View Model (MVM) has been proposed using which the method engineer conceptualizes his/her method. The MVM is shown in Figure2. A *thing* is an abstraction of the set of concepts of the method. These are partitioned into *Product Entities, Link* and *Constraint* to identify concepts that are for product construction, connecting two product entities together and constraint enforcement respectively.

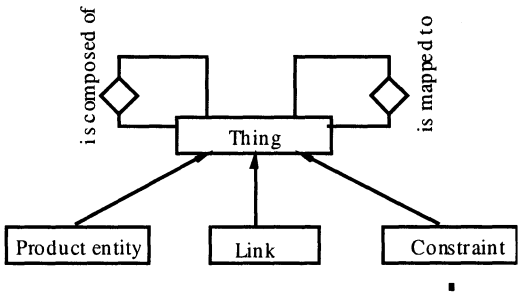


Figure 2: The Method View Model

Things are related to one another by the *is composed of* and *is mapped to* relationships respectively. The *is mapped to* relates together things of two different models. It says that a thing of one model maps to that of another. The *is composed of* says that things of a product model are built out of simpler ones.

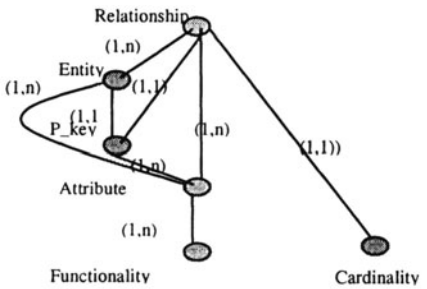
If s_i *is composed of* s_j then we shall refer to s_i as super concept and to s_j as sub-concept. The *is composed of* relationship is an M: N relationship whose cardinality is expressed in two attributes called m_min and m_max, respectively. These provide values for the minimum and maximum number of sub-concepts needed by a super-concept. For example if a relationship *is composed of* at least two entities then m_min=2 and m_max=n. That is, a minimum of 2 and a maximum of n entities can participate in a relationship.

The *is mapped to* specifies the correspondence between the *things* of two different models. Given two methods M1 and M2, if s_i *is mapped to* s_j, then s_i maps to s_j where s_i is a *thing* of M1 and s_j is a thing of M2. The *is mapped to* is an M: N relationship. For example in a mapping of an ER model to a Relational model, an Entity of ER model *is mapped to* a Relation of relational model.

3.1 *Is composed of* hierarchy

The *is composed of* relationships between the *things* of a method helps in building an *is composed of* hierarchy. Nodes of the hierarchy are the *things* and its edges are the *is composed of* relationships. Edges of the hierarchy have attributes min and max associated with them. The hierarchical levels are numbered with the bottom level as zero and

increases in the upward direction. We take the *is composed of* information in the same format as (Gup01), i.e. <super concept, sub concept, min, max> and convert it into a hierarchy.
As an example, consider the following list of things C and the *is composed of* relationships



ER Model

Figure 3: The *Is composed of* Hierarchy

C = < Relationship, Entity, Attribute, P-key, Functionality, Cardinality>

Is composed of **Relationships:-**

< Relationship, Attribute, 1,n >
< Relationship, Entity, 1, n >
< Relationship, Cardinality, 1,1 >
< Relationship, P-key 1,1 >
< Entity, Attribute, 1,n >
< Entity, P-key 1,1 >
< P-key, Attribute 1,n >
< Attribute, Functionality 1,n >
This results in the hierarchy of Figure 3.

4. DEVELOPING THE MAPPING

We derive the *is mapped to* relationships between two methods from their *is composed of* hierarchies. Given these hierarchies three cases arise (a) all mappings between nodes are given (b) only a subset of the mapping between nodes is given and (c) no mapping knowledge is available. Clearly (a) is a trivial case.

Once the derivation process starts it produces the *is mapped to* relationships with no further intervention from the method engineer. The relationships produced are presented to the method engineer who either accepts them or modifies them to meet his/her needs. Thus, in cases (b) and (c) the derivation process produces a feasible mapping for the method engineer. However, in comparison to case (a) for case (b) the derivation process produces an output, which is likely to be more correct and more acceptable

In general, the number of levels in the two hierarchies may be different: one hierarchy may have n levels and the other m, m<>n. Similarly, the total number of nodes in the two hierarchies may be unequal. Further these may be distributed across levels differently. For example, one hierarchy may have a total of p nodes and the other may have q nodes. Additionally, level 1 of the first method may have k nodes and the level 1 of the other may have j nodes. We can establish a mapping between the nodes of the two hierarchies in two ways:

(a) Establish node mapping directly or

(b) Break up node mapping into two

- Establish mapping between the levels of the two hierarchies. For example, this step can establish that the level m of one maps to level n of the other. We will refer to the pair (m, n) as **mapping levels**.
- Once mapping levels have been established we determine the mapping between nodes in the two hierarchies. This is done by taking a mapping level say (m, n), and determining the nodes of these levels that map to one another. For example, node k of m could map to node l of n for a mapping level (m, n). We will refer to (k, j) as **mapping nodes.**

We adopt approach (b) here.

4.1 Establishing Mapping between Levels

The underlying assumption in level mapping is that if a pair of mapping levels (p, q) is known then (p±1,q±1), (p±2,q±2) are all mapping levels. This follows from the fact that (p+1) *is composed of* p and (p-1) *is a component of* p. Similarly, (q+1) *is composed of* q and (q-1) *is a component of* q. If the levels in one of the hierarchies are

exhausted then the remaining levels above/below, if any of one *is mapped to* the topmost/bottommost level of the other. This further leads to the cases summarised below.

4.1.1 Case 1: Node mapping not given

In this case the level mapping between the two hierarchies is completely unknown. We assume that the 0^{th} level of M1 *is mapped to* the 0^{th} level of M2 (0,0). Starting from the (0,0) level, we establish mapping between each level above it till the levels in one or the other hierarchy is exhausted.

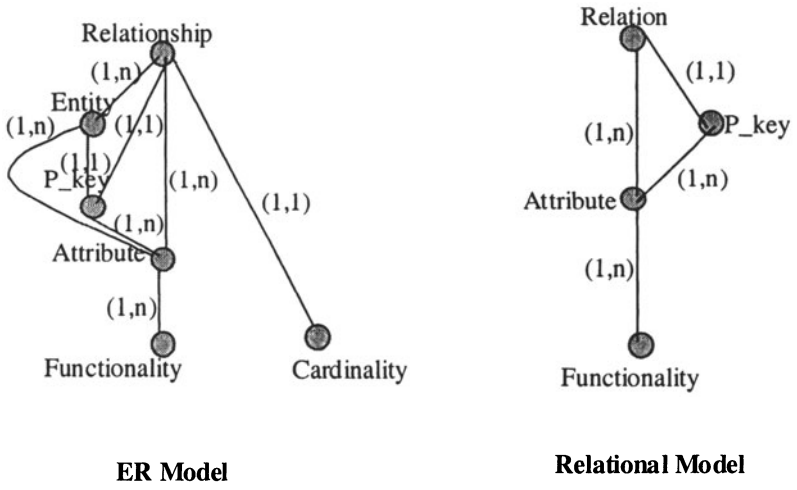


Figure 4: Hierarchies of ER and Relational Models

Thereafter, all the remaining levels, if any, of one are made to map to the topmost/bottommost level of the other. For example consider mapping of an ER model to a Relational model. The two hierarchies are shown in Figure 4. The mapping levels established between the two models when no node mapping input is given are as follows: - (0,0), (1,1), (2,2), (3,3), (4,3).

4.1.2 Case 2: Given Partial Mapping

In this case node mapping is partially known. The method engineer may supply one or more node mapping as input. These node mappings are used to determine mapping levels. We assume that if a pair of nodes map then their levels also map. This implies that if there is a pair of mapping nodes (p, q) then (m, n) is a mapping level and all the nodes at m map to the nodes at n.
For example, given A1 and A3 at levels 1 and 3 of one hierarchy and C2 and C5 at levels 3 and 5 of the other are the mapping nodes (A1, C2) and (A3, C5) then the mapping levels are (1,2) and (3,5).

If exactly one node mapping is given, then there is exactly one mapping level specified by the method engineer. For levels above we establish level mapping as for case 1. Additionally, the same procedure is followed for levels below till one or the other hierarchy is exhausted. All remaining levels then map to the bottommost level of the other. As an example, consider the mapping of levels of M1 and M2 as shown in Figure 5. The given input translates to the level mapping (l_{12}, l_{24}). Then, the mapping levels above (l_{12}, l_{24}) are (l_{13}, l_{25}) and (l_{14}, l_{25}). Now, we proceed to map the levels below (l_{12}, l_{24}). This yields the following mapping (l_{11}, l_{23}). (l_{11}, l_{22}) and (l_{11},l_{21}).

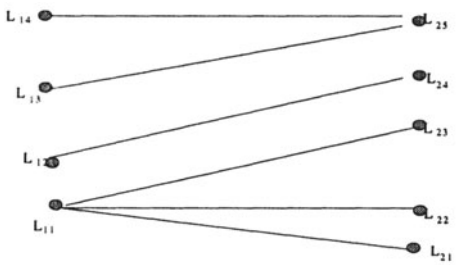


Figure 5: Given Partial mapping with single input

If more than one-node mappings are given, then the mapping level input thus determined should not consist of cross mapping levels as shown in figure 6.

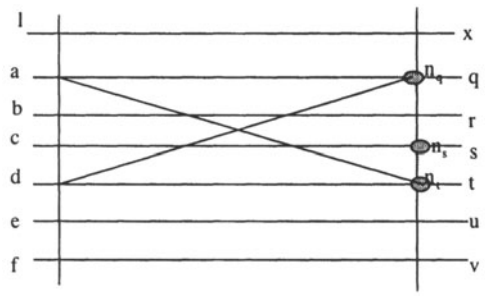


Figure 6: Cross level mapping

From the *is composed of* hierarchies of the two models the concept n_q which is a super concept of n_t must go above level a. If not then the *is composed of* relationship is violated. Similarly if n_t has a super concept at levels r and s and they map to levels above a in M2, then the super concept n_s of n_t becomes a super concept of n_q. Again a violation of the *is composed of* relationship occurs. Also, if n_q *is composed of* n_t then n_q goes to n_d is a violation. These cases are not practically possible.

If valid multiple node mappings are given, then starting from the lowest mapping level in the hierarchy, for every pair of levels, level mapping is established for the levels between that given pair as in case 1, till the levels of one or both the hierarchies is exhausted. There is a possibility that one or more levels still remain to be mapped between the given pair of levels. These remaining levels map to the topmost level of the given pair of levels. Further, it may be the case that there are additional levels in the hierarchies above the highest given mapping given levels. In that case, starting from the highest level the procedure of case 1 is adopted. Finally, there may be levels below the lowest given mapping level. As before, starting from this lowest level we proceed downwards till all the levels of one are exhausted. Any remaining levels are mapped to the bottommost level of the exhausted hierarchy.

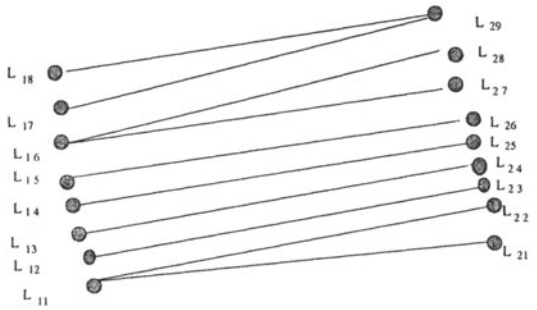


Figure 7 : Given Partial mapping with multiple input

As an example consider Figure 7 Given: - (l_{12}, l_{23}) and (l_{14}, l_{25}) and (l_{16}, l_{28}).
For the pair of levels (l_{12}, l_{23}) and (l_{14}, l_{25}) we get the mapping (l_{13}, l_{24}). The levels in both the hierarchies between this pair of levels is exhausted so, we proceed to the next given pair of levels. (l_{14},l_{25})and (l_{16},l_{28}). As earlier we get the mapping (l_{15}, l_{26}) and (l_{16}, l_{27}). Further, there are levels above the given mapping levels. These result in (l_{17}, l_{29}) and (l_{18}, l_{29}). The levels below the lowest given mapping are mapped similarly and we get (l_{11}, l_{22}) and (l_{11}, l_{21}).
We apply this to our hierarchies of Figure 4. Let the given mapping levels be (1,1) and (3,3). Then, we get (2,2). Now, the levels of the Relational model are exhausted after the mapping of the levels (3,3) but still some levels remain in the ER model for mapping. Then, adopting the procedure of case 1 we get (4,3). Below (1,1) we still have some levels remaining in both the models. We proceed to map them to obtain (0,0). Both hierarchies are exhausted now and the process terminates.

4.2 Establishing Mapping between the Nodes of Mapping Levels

In establishing node mapping the underlying assumption is that if the *is composed of* relationships of the node of one model are **similar** to the *is composed of* relationships entered into by another node of the other model at the mapping level, then the two nodes are mapping nodes. In other words, if the sub hierarchy rooted in a candidate node is the same as that of the other, then there is a topological probability

that the two nodes map, provided the two nodes are at the same mapping level.

Consider the mapping level (j, k) of Figure. 8. Let j consist of nodes n_{j1}, n_{j2},…n_{jp}. Similarly, let k consist of n_{k1}, n_{k2},….n_{kq}. Let n_{ji} be a sub concept of n_{xg} and a super concept of n_{ha}. Then, look for a node n_{kl} which is a sub concept of a node n_{yh} and a super concept of n_{ia} at level y such that x, y and h, i are mapping levels. If min and max for "n_{xg} *is composed of* n_{ji}" are equal to those for "n_{yh} *is composed of* n_{kl}" and the min and max for "n_{ji} *is composed of* n_{ha}" are also equal to those for "n_{kl} *is composed of* n_{ia}" then we say that n_{ji} maps to n_{kl}. This node mapping is expressed as (n_{ji}, n_{kl})

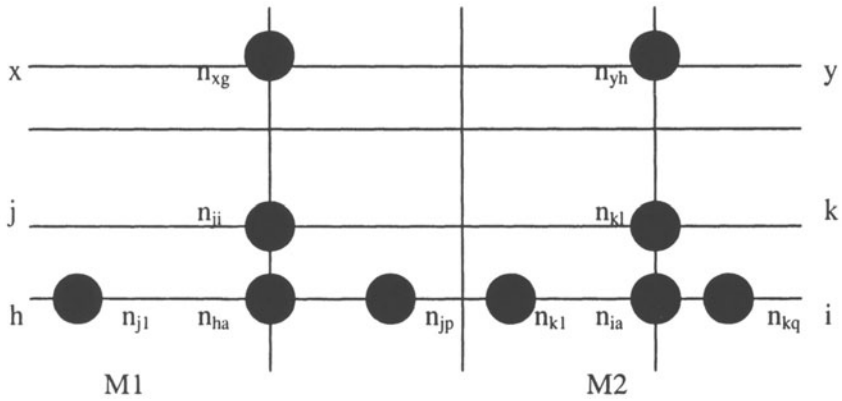


Figure 8: Level mapping of super concepts for a node.

We define a "similarity" between two *is composed of* relationships represented as the *is composed of* links in the hierarchy as: A **similarity** exists between these provided

1. Levels of the super concepts and the sub concepts of these nodes are mapping levels and
2. Min and Max of the *is composed of* relationship of these nodes with their super concepts and sub concepts respectively are the same

When, the two *is composed of* relationships are not similar, then we say they are dissimilar. The greater the number of similarities, the greater is the level of confidence that the two nodes map to each other. The maximum amount of confidence in node mapping is when there are only similarities and no dissimilarities. The least amount of

confidence is when there are all dissimilarities and no similarities and there are varying shades of confidence in between.

We propose that node mapping should be considered only for the cases, whose number of similarities is equal to or more than the threshold value. We define this threshold value as follows:-

***Threshold**: (Number of is composed of links in the sub tree) / 2*

As an example consider the five cases given in Table 1. All of these shows a node with 6 *is composed of* relationships. Consequently, the value of threshold is 3. The first case of Table 1 is shown in Figure 8. Here all the *is composed of* links of n_{ji} and n_{kl} at mapping level (j, k) are similar. The similarity exists between the following links: (A, H) (B, G) (C, I) (D, J) (E, K) and (F, L). As there are six similar *is composed of* links we have six similarities. Additionally, there are no dissimilarities. This is the best case with all similarities and no dissimilarities.

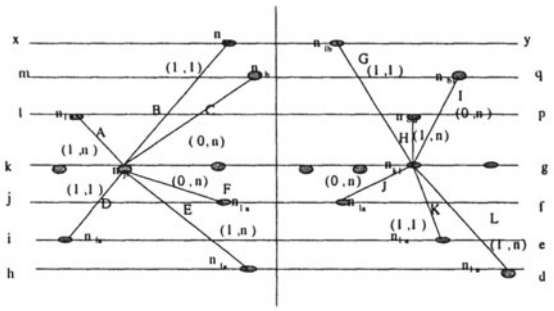


Figure 9: Node mapping

In the second case of Table 1 there are five similarities and one dissimilarity. As this lies above the threshold value this case is acceptable for node mapping. In the third case of Table 1 there are four similarities and two dissimilarities. This case lies above the threshold value and is acceptable for mapping. The fourth case shows equal number of similarities and dissimilarities. This case is equal to the threshold level, hence it is also acceptable for node mapping.The fifth case has no similarities and six dissimilarities, which are the maximum possible dissimilarities between two nodes. This is the worst case for node mapping and is rejected.

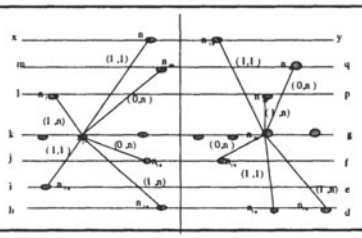
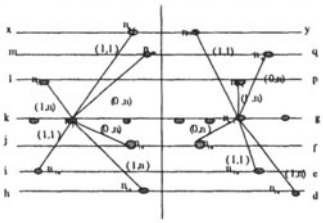
4.2.1 Presenting the Node Mappings

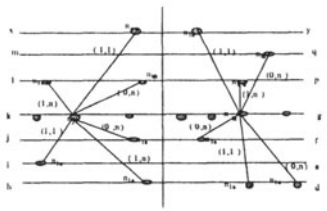
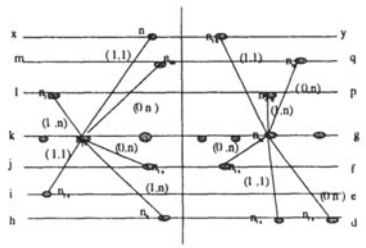
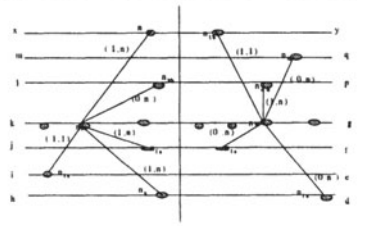
Once the similarities are determined and feasible node mappings are obtained then the proposed node mappings are displayed for possible modifications by the method engineer. This display is in the form of a table, which has one row for every node mapping and each row has four entries: node of one method, node of other method, number of similarities/ number of *is composed of* links, threshold value. The first two entries of each row identify the nodes that are mapped to each other. The third and fourth entries give an indication of the confidence in the node mapping.
For example,

n	m	4/6	3

Says that n maps to m. Out of 6 *is composed of* links, only 4 are similar and the threshold value is 3. The level of confidence in this case cannot be very high since the separation between the level of similarities and the threshold value is only 1 which is rather small.

Table 1: Level of Confidence in node mapping

Si.No	No. of Similarities /Max. No of Similarities	Accept-ability	Figure
1	6/6	Yes	
2	5/6	Yes	

3	4/6	Yes	
4	3/6	Yes	
5	0/6	No	

5. MAPPING OF RELATIONAL MODEL TO XML

For purposes of illustration we will consider the cases where no node mapping is given. Let the things and the *is composed of* relationships of the Relational model be as follows: -

Given: -
(Things of Relational Model)
C = < Relation, Attribute, key, functionality>
(*Is composed of* Relationship of Relational model)
< Relation, Attribute, 1,n >
< Relation, Key, 1, 1 >
< P-key, Attribute 1,n >

Similarly, let the things and the *is composed of* relationships of the XML model be as follows: -

(Things of XML Model)

C=< Containment,Element, Attribute, ID, Multiplicity, Functionality>

(*Is composed of* Relationship of XML Model)

< Containment, Element, 1,n >
< Containment, Multiplicity 1,1 >
< Containment, ID, 1,1 >
< Containment, Attribute, 1,n >
< Element, Attribute 1,n >
< Element, ID 1,1 >
< ID, Attribute 1,n >
< Attribute, functionality 1,1 >

The *is composed of* hierarchies are shown in Figure 10 below.

Step1: -

The mapping is assumed as (0,0).

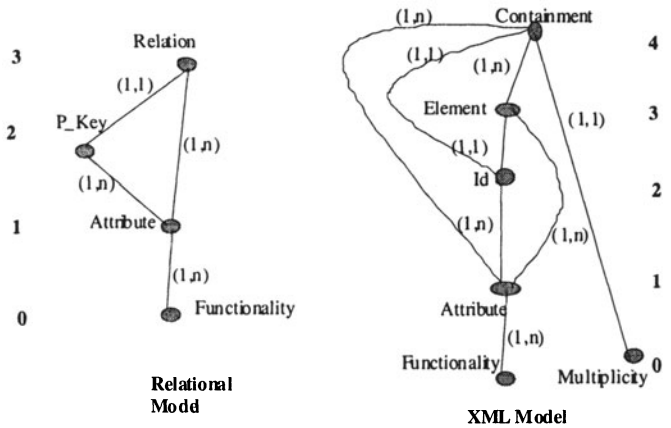


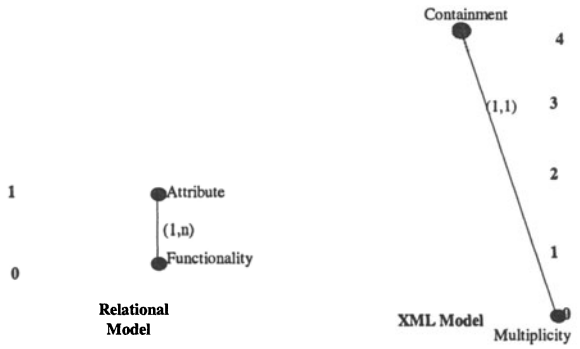
Figure 10 : *Is composed of* hierarchies of the Relational and XML models

Mapping levels are established as in Section 4 by mapping the levels above the given level (0,0) of the two models one by one. Giving us (0,0), (1,1), (2,2), (3,3).

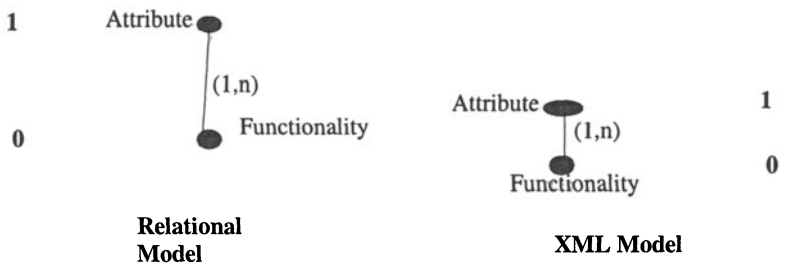
Now the levels of the Relational model are exhausted but level 4 in XML is still to be mapped. It *is mapped to* the topmost level of the Relational model. So we get (3,4). The next step is node mapping at these mapping levels.

Step2: -

For level (0,0) we select Functionality in Relational model and look for a mapping node in XML. There are two nodes Multiplicity and Functionality in XML.



The pair (Functionality, Multiplicity) has no similar *is composed of* links and is rejected for mapping. The sub hierarchy for this pair of nodes is shown above.

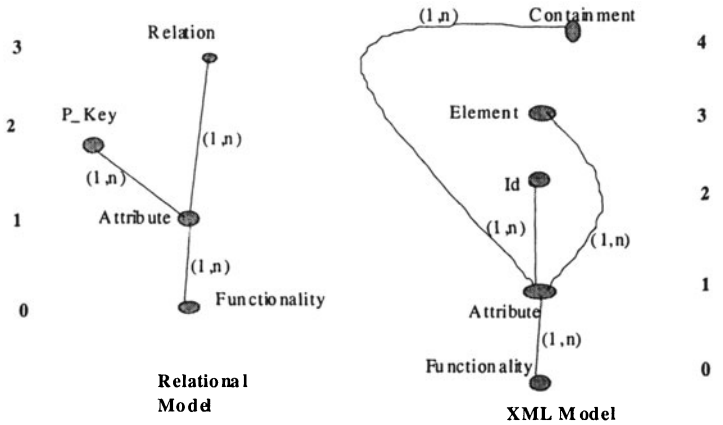


For the other pair (Functionality, Functionality) there is only *one is composed of* link in each of the sub hierarchies. These links are similar because

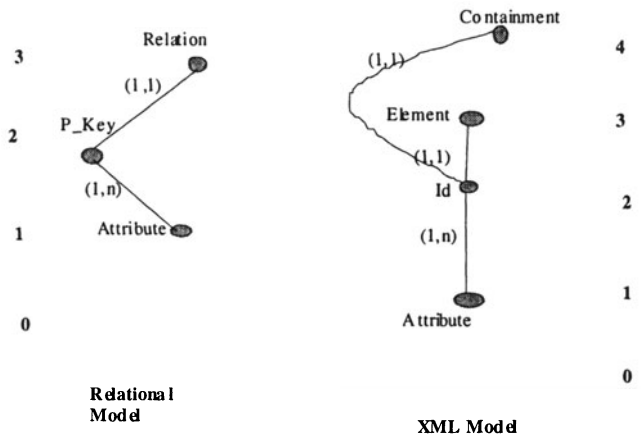
1. Min, max of the two are the same and
2. The levels of the super concepts are at mapping levels.

As there are no dissimilarities this is the case of maximum confidence and we get the node mapping (Functionality, Functionality).

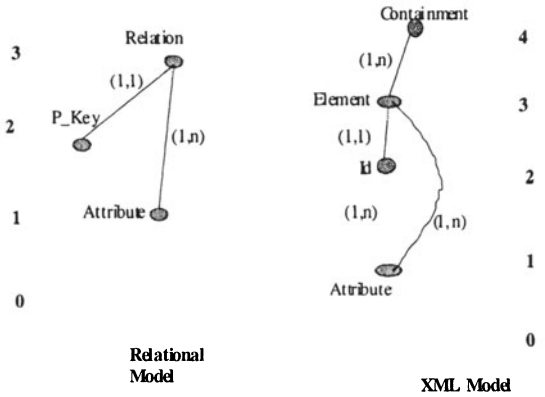
The next mapping level (1,1) has the pair (Attribute, Attribute). As shown in the figure there are three similarities and one dissimilarity. This lies above the threshold and we get the node mapping (Attribute, Attribute).



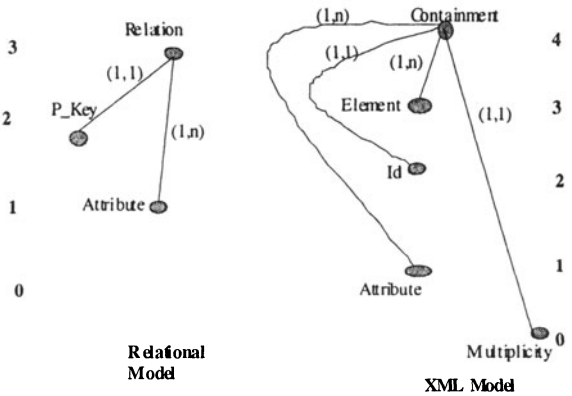
The third mapping level has (P-Key, Id) nodes. There are two similarities and one dissimilarity. This lies above the threshold level and is acceptable.



The fourth mapping level has (Relation, Element). This has two similar and one dissimilar *is composed of* link. Hence this mapping is acceptable.



The last node (Relation, Containment) has two similar and two dissimilar links. Since it lies at the threshold it is also acceptable.



These mappings are presented to the method engineer in Table 2.

Table 2 : Proposed Node Mappings

Relational	XML	Similarity	Thresh-old
Functionality	Functionality	1/1	0.5
Attribute	Attribute	3/4	2
P_key	Id	2/3	1.5
Relation	Element	2/3	1.5
Relation	Containment	2/4	2

6. LIMITATIONS

Our approach for determining mapping has the limitation that it does not work for *is composed of* hierarchies that contain a mix of concepts, passive as well as active. Examples of active concepts are the notions of operators, processes, events, etc. When these are mixed with passive concepts like entity, relationship etc. then it becomes difficult to determine level mapping. Consider OM and FM of OMT that contain the notion of an operator and process respectively. The *is composed of* hierarchies of these are shown in Figure 11. Let the given level mapping be (process, operation) shown by the dotted line. Then, by case 2 of section 4.2 we obtain the following level mappings: -

(2,0), (1,0), (0,0), (2,1), (2,2)

This mapping of level shows that the things at these levels are also mapping, which means that the things at levels (2,1,0) of OO Model map to (2,1,0) of DF model giving a probable node mapping ((Process, Data _flow, Store, Input, Output, Operation, Attribute), (Association, Object., Multiplicity, Operation., Attribute)) from mapping level (0,2).

It can be seen that the mapping exists between (Association, Process). This mapping is semantically meaningless. Thus, we assume that our *is composed of* hierarchies contain concepts that are either all passive or all active. If the concepts of operation and process as well as their *is composed of* relationship are removed from Figure.11, then it can be seen that the semantic difficulty does not arise.

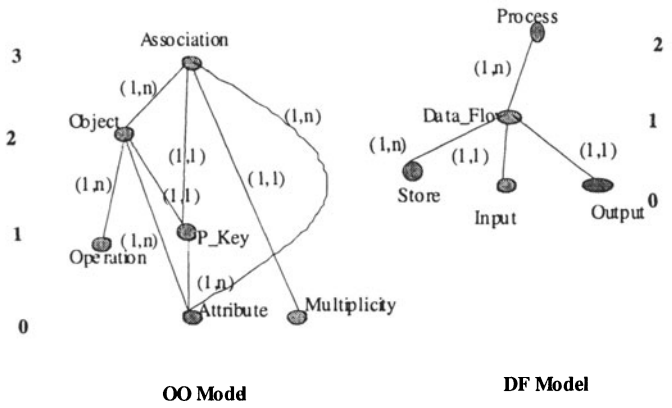


Figure 11: *Is composed of* hierarchy of OM and DF

Separate hierarchies for the active concepts need to be built and the process described in this paper applied to these as well. The active and passive mappings yield the complete mapping between the models.

7. CAME SUPPORT

The CAME support to generate compound methods is shown in Figure 12. We view this figure in two parts, the top part and the bottom part. The former is for generating atomic methods as developed in (Gup01) whereas the latter is for generating the *is mapped to* relationship and applies to compound methods. We consider the second of these here.

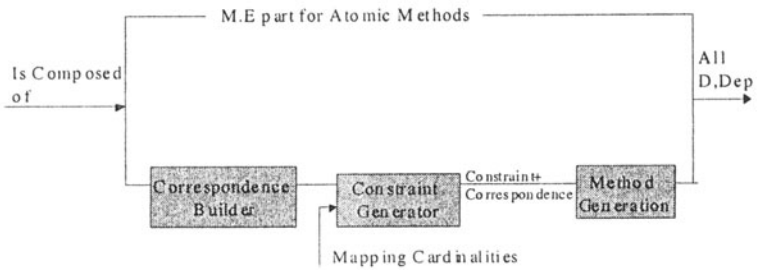


Figure 12 : The Complete Came Tool

The ideas of this paper form the first component, called Mapping Builder. This component produces a feasible mapping and displays it to the method engineer. This establishes that one concept *is mapped to* another. Now, the cardinality of this mapping has to be specified. Just as done in (Gup01), we expect the CAME tool to produce the constraints of completeness, conformity, and fidelity from the cardinality and mapping information. This is done by the Constraint Generator component of the CAME tool.
The generated constraints together with the mapping information are then input to the Method Generator. This generator produces the set of decisions and dependencies that are needed to define method statics.

8. CONCLUSION

We have experimented with our approach with a number of methods both constructional and transformational and the results are rather encouraging. It seems to us that the essential reason the approach works is because the concepts of a method are composed of one another. Thus, starting from simple concepts, complex method concepts can be built. The *is composed of* hierarchy captures this 'composed' nature of a method well. Our approach works for semantically meaningful hierarchies: those that have purely passive or active concepts in them.
A prototype of the Mapping Generator is already running in our laboratory. We are now working on the Constraint and Method Generator modules of the CAME tool. Thereafter, we will consider the schema mapping process where our main thrust is in producing an optimal process.

9. REFERENCES

[Mcb'98] P.McBrien, A.Poulovassilis, A general framework for schema transformation. Data and Knowledge Engineering 28 (1998) pp 47-71.

[Mcb'97] P.Mcbrien, A.Poulovassilis, A formal framework for ER schema transformation in Proc. of ER'97, vol.1331, LNCS, 1997, pp.408-42.

[Gup 01] Gupta D. Prakash N, Engineering Methods from their Requirements Specification, Requirements Engineering Journal, 6, 3, pp133-160.

[Pra 97] Prakash N. Towards a formal definition of Methods, Requirements Engineering Journal 2,1, pp 23-50.

[Wil 01] K.Williams, M. Brundage, P.Dengler,J.Gabriel, A.Hoskinson, M.kay, et al Professional XML databases.

[Ram97] J.Rambaugh, M.Blaha, W.Premerlani, F.Eddy, W.Lorensen, Object Oriented Modeling and Design, Prentice Hall of India 1997.

[Kor 91] H.F.Korth, A.Silbershatz, Database System Concepts, McGraw Hill 1991.

[Chen 97] Yangjun Chen and Wolfgang Benn, Rule-based Technology for Schema Transformation, Proceedings of the 2nd IFCIS International Conference on Cooperative Information Systems (CoopIS '97).

[Hal 95] T. A. Halpin and H. A. Proper, Database Schema Transformation & Optimization, OOER'95: 14th International Conference on Conceptual Modeling, Springer, LNCS, Vol. 1021, pp. 191-203.

[Ren 01] Renguo Xiaou, Tharam S. Dillon[1], E. Chang, and Ling Feng, Modeling and Transformation of Object-Oriented Conceptual Models into XML Schema, LNCS 2113, p 795.