Knowledge Sharing and Inconsistency Checking on Multiple Enterprise Models

Yun-Heh Chen-Burger
Artificial Intelligence Application Institute,
The University of Edinburgh,
80 South Bridge, Room E32, Edinburgh EH1 1HN, UK
email: jessicac@aiai.ed.ac.uk
Tel: +44 131 650 2756, Fax: + 44 131 650 6513

Abstract

The approach of *Multi-Perspective Enterprise Modelling* is now more commonly accepted and used in practice as a way to manage knowledge than ever before. However, the concept of applying multiple modelling languages to describe the same domain may still sound frightening to many. In addition to the cost, time and complexity involved, problems such as knowledge sharing between multiple models and achieving and maintaining integrity between them are also important. We argue that *Multi-Perspective Enterprise Modelling* is helpful and in some situations necessary. This paper gives examples of how formal methods, such as logical languages, can provide assistance in making such an approach more appealing and transparent. We suggest that the *MPM* approach is valuable in representing, understanding and analysing a complex domain, but that much automated support is needed.[1][2]

Key-words Knowledge Management, Knowledge Sharing, Multi-Perspective Modelling, Business Modelling, BSDM, Enterprise Modelling, Process Modelling, Knowledge-Based Support Tool, Business Process Re-Engineering, Role Activity and Communication Diagram.

[1]This paper reports a part of work carried out under AOEM project. The AOEM project is sponsored by the Defense Advanced Research Projects Agency DARPA under the contract number N66001-99-D-8608. The U.S. Government and the University of Edinburgh are authorised to reproduce and distribute reprints for their purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either express or implied, of DARPA, the U.S. government or the University of Edinburgh.

[2]This paper will appear on International Joint Conference on Artificial Intelligence, IJCAI, Knowledge Management and Organizational Memories Workshop and its proceedings at Seattle, Washington, August 2001

1 Introduction

Today's economy is often called a *knowledge economy* indicating that stake-holders of the right kind of knowledge may gain competitive advantages over its rivals. This knowledge may be roughly divided into two types: internal and external knowledge. Internal knowledge is the *corporate knowledge* within an organisation and external knowledge is about the economic environment the organisation operates in.

A modern enterprise is often a virtual entity which consists of many sub-organisations which are distributed across different geographical areas, each possessing different expertise and specialising in certain functions. This complicates the task of treating *corporate knowledge* as a whole and making effective use of it. Furthermore, the economy in which an organisation operates is very dynamic which requires an organisation to react appropriately and promptly — in adapting their goals and processes. This paper focuses on the capture of *corporate knowledge* using *Multi-Perspective (Enterprise) Modelling (MPM)* techniques.

Example enterprise modelling languages are *IBM's BSDM Business Modelling Language*[11], Ould's *Business Process Modelling* language[17], Dobson and Strens's Organisational Modelling language[13], Fox and Gruninger's ontology based enterprise modelling[8], *Eriksson* and *Penker's* business modelling based on extensions of *UML*[7], and *IDEF* methodology's process modelling languages such as IDEF3[15] and IDEF0[16]. In this paper, we distinguish between (enterprise) modelling languages and models: a modelling language is the language that has been used to describe a domain; the description of the domain using that modelling language is the end product of a modelling exercise, we call the created product *Enterprise Models*.

Multi-Perspective Modelling (MPM) techniques make use of multiple modelling languages to describe a domain which allow one to present and analyse organisational knowledge from different points of view, which in turn allows the knowledge to be used for different purposes. The *MPM* approach is necessary because organisational knowledge is often so complicated and of heterogeneous types that normally no single modelling method can capture all of the important aspects and present them clearly and appropriately. Thus a Multi-Perspective modelling approach makes use of multiple modelling languages which complement each other and work as a whole to describe the enterprise knowledge better.

Furthermore, the required uses of the produced *Enterprise Models* are often diverse: businesses often need to examine different but specific aspects of the knowledge for different purposes as a part of their decision making process. This knowledge may come from the same body of knowledge, but just examining the concerned sides of it at a time. This demands a diversified but specialised presentation of the *corporate knowledge* which allows analysis to be performed on the desired aspects of the knowledge for each specific purpose.

The components of domain knowledge of an enterprise model, however, can be highly inter-dependent if they are not structured and presented appropriately. It is therefore important to have a mechanism which allows the relevant information to be gathered and presented in a clear, concise and structured way which is not overburdened with other irrelevant information. Since a multi-perspective modelling approach uses several different modelling languages, each language provides a specialised presentation to the knowledge domain which also allows insight analysis into the specific aspects of the domain.

The *MPM* approach is already used in research and practice: Common KADS methodology [18] embodies several modelling languages to help understand and capture domain knowledge and to help the design of knowledge based systems; Booch, Rumbaugh and Jacobson [1] fully embrace this approach and have offered a suite of inter-supportive modelling notations as part of the *Unified Modelling Language*, for gathering requirements and development of software systems; Frank[9] advocates this approach based on which multiple notations have been used and a multi-perspective knowledge management system (MEMO) was developed; Zachman's[21] Framework for Enterprise Architecture, suggests using a variety of modelling languages to capture and describe the different aspects of a domain. The importance and benefits of using multiple and complementary modelling languages to represent a complex knowledge body is well-recognised and adapted more frequently than before.

During the *Air Operation Enterprise Modelling (AOEM)* project[14], a Multi-Perspective Modelling approach was taken. The domain of (military) Air Operations is complex. A main source of knowledge regarding Air Operations was provided to the initiative in an *IDEF0* model[3]. It consists of 290 functions, 307 inputs (data types which provide input information for the functions), 294 outputs (data types or results which are produced by the functions), and 45 controls (data types which provide principles, guidance and information for executing the functions). In addition, documents written in natural language, informal diagrams and tables are provided — describing different parts of the air operations. This above information is aided with face-to-face explanation and email correspondence with domain experts. Several aspects are considered: the infrastructures used during the operations, the operations to be carried out, people involved and their actions and the interactions between them, policies that are followed, resources and information needed, and issues such as timing for cooperation during the operation.

[3]The Air Operations IDEF model was developed by Larry Tonneson, Zel Technologies, LLC, USA.

To illustrate these aspects, three types of models are initially chosen and built: a *Domain-Model* provides a taxonomic structure to capture all the high-level and fundamental concepts that are important to the air operation, a *Business Model* to capture the infrastructure involved in the operations and the detailed concepts that are used in the context of air operations, a *Role Activity and Communication Model* to identify the type of actors who are involved in the operations, their individual operations and the interactions between them.

Although these models were appropriate to the needs of the project, we commonly faced the problem, as any other *MPM* initiatives would, of keeping track of information that is distributed and shared between different models and to make sure that this is consistently represented in all models. Furthermore, on the higher level of abstraction, the "principles of business operations" that are described, subscribed and implied in all models must also be consistent with each other. To obtain and maintain such consistency is a highly labour-intensive task and can be error-prone when no computing aid is available.

This paper describes a part of our work carried out under the *AOEM* project. It reports our attempts in providing a framework that are generic across application domains and appropriate for the MPM approach. It also provides a part of the initial work carried out for the *IRC AKT* project.[5]

Examples of how logical methods can provide automatic or semi-automatic support to obtain and maintain consistency in an MPM approach are described in detail in the later sections. We firstly give an overview of our multi-perspective modelling approach.

2 Overview of Multi-Perspective Modelling Framework

To maximise the advantages of the *MPM* approach proposed in this paper, a few principles are followed. Firstly, all of the modelling languages that are chosen to build the enterprise models must be suitable to describe the chosen problem domain and appropriate to achieve the modelling objectives. Secondly, the chosen modelling languages should be complementary of each other so that all of the concerned knowledge are described among them. Thirdly, these modelling languages should be "compatible" with each other, i.e. their modelling principles are sufficiently similar to each other so that the built model can achieve a consistent and coherent view of the domain.

It is also important that most of the models are built based at the same (or at least similar) level of abstraction: if one (or more) of the modelling language allows multiple levels of abstraction, e.g. using the modelling languages of *IDEF0*[16] or *IDEF3*[15], appropriate guidelines must be established to determine which level of abstraction is mapped to the other non-decomposable modelling languages.

According to Visser, Jones, Bench-Capon and Shave's[20] four categories of heterogeneity existing between bodies of information: the *paradigm*, *language*, *ontology* and *content heterogeneity*, the MPM approach described here deals with a set of models that have *language* and *content heterogeneity*, since those models are written in different modelling lan-

guages (language heterogeneity) and may describe different parts and aspects of the same domain (content heterogeneity). Since enterprise models may be used to provide partial specification documents for developing software systems, the same paradigm will normally be chosen (if any) and followed for all models, e.g. the Object-Oriented or Entity-Relational paradigm. Enterprise models may also be independent of the software systems that are built based on them, therefore, independent of any software paradigm. The MPM approach given here does not try to resolve the *ontology heterogeneity* – it assumes that this issue has already been dealt with and that a common ontology has already been formed – the enterprise models merely reflect the coherent vision and consensus of an organisation. Discussions about modelling approach and concept mapping principles between models in a MPM initiative are given in more detail in [3].

Figure 1 shows our *MPM* approach. As mentioned earlier, three models are used: *IBM BSDM's Business Model (BM)*[12][4][11], *Domain-Model (DM)*[2], and *Role Activity and Communication Diagram (RACD)*[14].[5] Each of the three circles represents the domain knowledge that is covered by each model. The overlapping areas denote the common knowledge that is covered in different models, although it is presented in different forms (i.e. using the specialised model primitives) in each model. The area that is covered by only one model denotes the specialisation of the particular modelling language that describes the kind of knowledge that are not (or can not be) captured by any other models. An example of such specialised knowledge is the type of "role" that people play in an air operation as well as its responsibilities, operations and interactions with other roles which is only covered by *RACD*. The Domain Model may cover all of the overlapping areas between BM and RACD. Whether what is considered to be essential and fundamental to the domain and is therefore captured in the DM is a design decision. Figure 1 presents one case of DM.

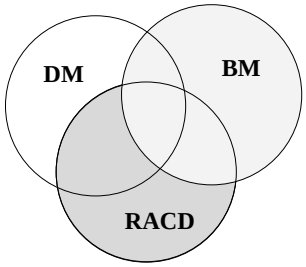

Figure 1: Overview of Multi-Perspective Modelling Approach

Figure 2 depicts how the *Domain-Model (DM)* has been used as a backbone, a *light-weight ontology*, for the *MPM* approach. It provides a taxonomic structure to store the fundamental and important knowledge of the domain. Two main types of knowledge have been captured: the higher-level classification information about the domain itself and the (lower-level) model concepts that are often represented using model-specific primitives in other models. Typically, an instantiation of a lower-level model concept has a direct correspondence to objects of the described domain area.

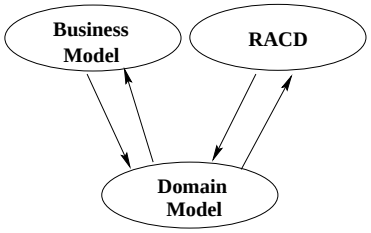

Figure 2: MPM using Domain-Model as a Backbone

Because the information stored in the *DM* is common and sharable between different models, it forms a natural communication media for knowledge transfer and translation. Model concepts that are described in one model are mapped to *DM* which are then mapped to model concepts that are captured in another model. Given the mapping of concepts, the appropriate knowledge can be mapped, shared and translated between models. This approach has been implemented in a rule-based system, *KBST-EM*[3], as a part of the *AOEM* project.

Note that although the Domain-Model is (logically) *typed* and similar to that of an object-oriented (OO) class model, it does not imply that any software system, that is built based on it, to follow the OO paradigm. The Domain-Model explicitly represents one relationship, is-a, between classes which will be used to check the semantics of different models are consistent. If an entity-relational view of the knowledge is desirable, an ER data model may be included as a part of the enterprise models; model concepts that are captured in the ER model can communicate with other models through the domain model. More details of the semantics of a *Domain-Model* and how model primitives and concepts are mapped between multiple models through *DM* are given in [2][6] and [3].

3 Obtaining and Maintaining Consistency and Integrity

Enterprise models are domain-specific because they are often used to describe a business domain that is within the operational context of an organisation; they are also method-specific, i.e. they are often described using a chosen modelling language following its standard practice to achieve a desired goal. Researchers have provided guidelines and mechanisms in the past to obtain and maintain the integrity and consistency among data.

The *Entity Relational Modelling* method often includes tasks of capturing domain-specific integrity constraints as a part of its *Data Modelling* exercises in which the integrity

[4] *BSDM* stands for *Business System Development Method.*

[5] *RACD* was developed by the author of the paper specifically to meet requirements for the AOEM project. It was adapted from the *Role Activity Diagram*[17] with its process notations extended with influences from *IDEF3*.

[6] Domain-Model is called Meta-Model in the reference.

constrains are checked against the design of schema before the actual database is implemented [6]. Thalheim[19] classifies such constraints in a three-layered hierarchical structure and describes them in a formal algebra. Domain-specific integrity constraints are sometimes called "business rules" which indicate the directives and policies that are followed in a business operation. To obtain the consistency between business rules, Grosof[10] suggests a prioritised conflict-handling mechanism for business rules based on logical inferencing techniques. Visser, Jones, Bench-Capon and Shave[20] illustrate and classify ontological mismatches and suggest guidelines as how these mismatches may be resolved. Various types of BSDM's business model method-specific consistency checking and error-correction advice are described in detail in [12] and [4].

All of the above describe the capturing and handling of integrity constraints for only one (modelling) method and of the same "aspect" of the information. This paper focuses on identifying generic integrity rules that are applicable across models (each model may be using different modelling languages) and that each of the models may cover different perspectives of the same knowledge body. The rationale is that although these models cover different prospectives of the domain, since they are describing the same domain, they should together present a consistent view of that domain.

The issues to be considered here are two fold: the similarity of model concepts that have been captured in different models and the compatibility of model primitives used in different models. It is the case that the more similar the model concepts and the more compatible the modelling primitives of different models are, the more plausible it is that integrity rules can be constructed and applied. The concept mapping mechanism that is described in the previous section tackles the issues of matching concepts between different models. Based on this mechanism the integrity-checking can be identified and performed on the mapped concepts.

A set of fundamental consistency rules are proposed. These consistency rules systematically and exhaustively search for all inconsistencies between models and present this information to the modellers. Although the consistency rules may provide error-correction advice, nevertheless, the final decision of whether or how these models are changed to be coherent with each other lies within the modellers' control.

We use A $\cong$ B to denote that model concept A is (conceptually) *fully equivalent* to model concept B, where A and B may be captured in different models and represented in different forms, i.e. using the appropriate model primitives in that modelling language. The fact that concepts A and B are *fully equivalent* is evidently shown in two aspects: i.e. they share the same name and the same definition (e.g. in natural language).

We use A $\doteq$ B to denote that model primitive A is *compatible* to model primitive B, where A and B are used in different modelling languages. To judge whether model primitives A and B are compatible, they must represent the similar function in their own modelling languages. Two types of inference operators of different strength of enforcement have been deployed: A $\Rightarrow$ B means that if A is true then B *must be* true, A $\triangleright$ B means that if A is true then B *may be* true.

The set of consistency rules is given below.

(1) Consistent Representation of Information

$$\begin{array}{l}\forall T1, M1, T2, M2, O1, O2, Att, Value1, Value2.\\ model\_primitive\_of(T1, M1) \wedge\\ model\_primitive\_of(T2, M2) \wedge\\ (T1 \doteq T2) \wedge\\ object\_type((O1, T1), M1) \wedge\\ object\_type((O2, T2), M2) \wedge\\ (O1 \cong O2) \wedge\\ object\_attribute\_in\_model((Value1, Att), (O1, T1), M1) \wedge\\ object\_attribute\_in\_model((Value2, Att), (O2, T2), M2)\\ \Rightarrow\\ Value1 = Value2\end{array}$$

where $model\_primitive\_of(T1, M1)$ indicates that $T1$ is a model primitive (type) of model $M1$; $object\_type((O1, T1), M1)$ defines that the model object $O1$ is of model primitive type $T1$ in Model $M1$; $object\_attribute\_in\_model((Value1, Att), (O1, T1), M1)$ stores the attribute value in $Value1$ for attribute Att for model object $O1$ in model $M1$. This formula indicates that **if** two model primitives, $T1$ and $T2$, in models, $M1$ and $M2$, are *compatible* and that the model objects, $O1$ and $O2$, of model primitive (type), $T1$ and $T2$, are *fully equivalent*, and that both objects, $O1$ and $O2$, have the same attribute Att, **then** the corresponding attribute values of $O1$ and $O2$, $Value1$ and $Value2$, *must be* the same.[7]

This consistency rule ensures that the information that is shared across models is consistently represented in the relevant models.

(2) Correct Specialisation of Concepts

$$\begin{array}{l}\forall T1, M1, T2, M2, O1, O2, S1.\\ model\_primitive\_of(T1, M1) \wedge\\ model\_primitive\_of(T2, M2) \wedge\\ T1 \doteq T2 \wedge\\ object\_type((O1, T1), M1) \wedge\\ object\_type((O2, T2), M2) \wedge\\ O1 \cong O2 \wedge\\ object\_type((S1, T1), M1) \wedge\\ sub\_concept(S1, O1, M1)\\ \Rightarrow\\ \neg\exists S2.object\_type((S2, T2), M2) \wedge\\ S1 \cong S2 \wedge\\ sub\_concept(O2, S2, M2))\end{array}$$

where $sub\_concept(S, O, M)$ denotes that model concept S is a specialisation or sub-concept of model concept O in model M. This formula imposes a consistent definition on the specialisation of concepts between models, i.e. **if** two model primitives, $T1$ and $T2$, in models, $M1$ and $M2$, are *compatible* and model objects, $O1$ and $O2$, in model $M1$ and

[7] The same attribute may also be given a different name in different models. We simplify this in the formula. Indeed, attributes may also only be "similar" and not "fully equivalent", but we restrict ourself to only "fully equivalent" attributes here.

$M2$, are *fully equivalent*, and that the model objects $S1$ is the sub-concept of $O1$, **then** it *must not* be the case that another concept $S2$ is found in model $M2$ which is conceptually fully equivalent to $S1$ and is the super-concept of $O2$.

This rule does not restrict the case when a concept has been correctly specialised by models in different ways. For instance, the concept "car" may be specialised in terms of its building structure in one model; but it may be specialised in terms of its functions in another model. This should be allowed, since each model is designed to describe the different aspects of the same domain – as long as these models are consistent with each other when the same concepts are mentioned. This allows each model to present a "partial" view of the same domain, but together present a coherent and more comprehensive view of the domain.

(3) Consistent Application of Dependencies

$$
\begin{array}{l}
\forall T1, M1, T2, M2, O1, D1. \\
model\_primitive\_of(T1, M1) \wedge \\
model\_primitive\_of(T2, M2) \wedge \\
T1 \doteqdot T2 \wedge \\
object\_type((O1, T1), M1) \wedge \\
object\_type((O2, T2), M2) \wedge \\
O1 \cong O2 \wedge \\
object\_type((D1, T1), M1) \wedge \\
depends\_on(O1, D1, M1) \\
\triangleright \\
\neg \exists D2.object\_type((D2, T2), M2) \wedge \\
D1 \cong D2 \wedge \\
depends\_on(D2, O2, M2))
\end{array}
$$

where $depends\_on(O, D, M)$ indicates that information that is represented in model concept O depends upon the "existence" of information that is represented in model concept D in the model M. The above rule states **if** two model primitives, $T1$ and $T2$, in models, $M1$ and $M2$, are *compatible* and model objects $O1$ and $O2$, in model $M1$ and $M2$, are *fully equivalent*, and that the information that is represented in the model objects $O1$ depends upon information that is described in model objects $D1$, **then** it *may not* be the case that another concept $D2$ is found in model $M2$ which is conceptually *fully equivalent* to $D1$, but it depends on the information that is stored in model concept $O2$.

The dependency relationship indicated above is generic, i.e. it may include different types of "dependencies" in different models where each dependency may have different strength in constraining the corresponding models. We, therefore, use the weak inference operator $\triangleright$ to include all of these cases and allow flexibility for model design.

The above rule enforces a consistent business practice that is described in different models (or indeed different parts of the same model). Figure 3 gives examples of where dependencies can be derived. Three examples are given: dependencies may be derived from a process model, data model and a business model, each are captured in the specialised model primitives. Figure (a) abstracts a structure that is commonly

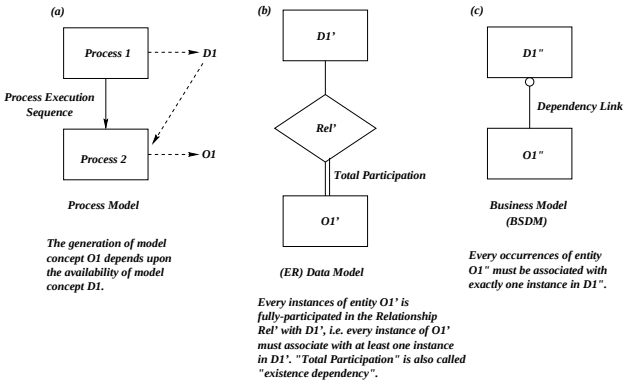

Figure 3: Example Dependencies in Different Models

seen in a process model[8] where the execution of process 2 is preceded by process 1 which produces information $D1$ that is used to generate information $O1$ by process 2. Based on such an architecture, one can derive that model object $O1$ depends upon model object $D1$, since instances of $O1$ can not be created unless instance(s) of $D1$ has (have) been created.

Figure (b) shows an example data model. In this case, we have used the notation from an *Entity-Relational* data model, but similar dependency relationships can be found in other types of data models. In the data model, a special type of relationship, *Total Participation*, has been used to indicate an *Existence Dependency*[6] which gives the constraint that every instances of entity $O1'$ must be related to at least one instance of entity $D1'$ in the relationship Rel'. Figure (c) specifies a similar dependency relationship that model object $O1''$ depends on $D1''$ in a dependency link[9]

These above examples suggest that dependency is a common property in many different modelling languages and that it can be extracted and generalised. Given the mapping of model objects between different models, the consistent application of dependencies can be checked across models.

The above consistency rule generalises all types of dependencies that may be described in modelling methods. Therefore, a weak inference (may be) has been used to allow modelling flexibility. In fact, there are many different types of dependencies, each may indicate a different level of "dependability". When application-domain-specific or method-dependent dependencies which have a strong implication on the described models are identified, the strong inference operator, $\Rightarrow$ (must be), can be used.

(4) Detecting Incompleteness

$$
\begin{array}{l}
\forall R1, M1, R2, M2, T1, T2, O1, P1, O2, P2. \\
relationship\_type(R1, M1) \wedge \\
relationship\_type(R2, M2) \wedge \\
R1 \doteqdot R2 \wedge \\
model\_primitive\_of(T1, M1) \wedge \\
model\_primitive\_of(T2, M2) \wedge
\end{array}
$$

[8]Example processes are those in the *IDEF0[16]*, *IDEF3[15]*, *RACD[3]* models.

[9]This notation is taken from *BSDM's Business Model[12]*.

$T1 \rightleftharpoons T2 \wedge$
$object\_type((O1, T1), M1) \wedge$
$object\_type((O2, T2), M2) \wedge$
$O1 \cong O2 \wedge$
$object\_type((P1, T1), M1) \wedge$
$object\_type((P2, T2), M2) \wedge$
$P1 \cong P2 \wedge$
$in\_relation(R1, O1, P1, M1)$
$\triangleright$
$in\_relation(R2, O2, P2, M2)$

where the predicate $in\_relation(R1, O1, P1, M1)$ specifies that model object $O1$ is associated with $P1$ in the relationship type $R1$ in model $M1$. The predicate $in\_relation$ is generic that it includes any kind of relationships that may be described between two model objects. This above rule states that **if** the relationships, $R1$ and $R2$, are compatible, and that model primitives $T1$ and $T2$ are compatible, and that model object $O1$ is fully equivalent to model object $O2$, and that model object $P1$ is fully equivalent to model object $P2$, and that model object $O1$ is associated with model object $P1$ in the relationship type $R1$ in model $M1$, **then** it *may be* the case that model object $O2$ is associated with model object $P2$ in the relationship type $R2$ in model $M2$.

Based on the appropriate mapping of relationships and model primitives between two models, this rule suggests a relationship, $R2$, in the second model $M2$ based on observations made on the first model. Since the predicate $in\_relation$ includes any relationships which makes the above rule quite generic, a weak inference, $\triangleright$ (may be), is therefore used. The rule is an example of what can be done to detect and *suggest* missing information in a model which contributes to the completeness analysis as a part of the consistency checking process. Similar principles can be applied to infer more specific types of information and more specific results may be concluded from that.

(5) Inferring Missing Information

$\forall T1, M1, T2, M2, O1, O2, Att, Value1, Value2.$
$model\_primitive\_of(T1, M1) \wedge$
$model\_primitive\_of(T2, M2) \wedge$
$(T1 \rightleftharpoons T2) \wedge$
$object\_type((O1, T1), M1) \wedge$
$object\_type((O2, T2), M2) \wedge$
$(O1 \cong O2) \wedge$
$object\_attribute\_in\_model((Value, Att), (O1, T1), M1)$
$\triangleright$
$object\_attribute\_in\_model((Value, Att), (O2, T2), M2)$

This formula is a weaker version of the consistency rule (1): *Consistent Representation of Information*, hence a weaker enforcement symbol has been used here, $\triangleright$. It indicates that **if** two model primitives, $T1$ and $T2$, in models, $M1$ and $M2$, are *compatible* and that the model objects, $O1$ and $O2$, of model primitive (type), $T1$ and $T2$, are *fully equivalent*, and that object $O1$ has an attribute Att with value $Value$, **then** object $O2$ in model $M2$ *may* also have the attribute Att with the value $Value$.[10]

This rule suggests that the information that is described in one model may also be usefully described in another model. However, since each model is meant to serve different aims, it is not necessary that all of the information of a concept is stored in all models even though this concept is captured in all models. This avoids flooding a model with excessive information. Because of this consideration, a weak inference symbol, $\triangleright$, has been used in this rule.

(6) Transitivity of Full Equivalence

$\forall O1, T1, M1, O2, T2, M2, O3, T3, M3.$
$model\_primitive\_of(T1, M1) \wedge$
$model\_primitive\_of(T2, M2) \wedge$
$(T1 \rightleftharpoons T2) \wedge$
$object\_type((O1, T1), M1) \wedge$
$object\_type((O2, T2), M2) \wedge$
$(O1 \cong O2) \wedge$
$model\_primitive\_of(T2, M2) \wedge$
$model\_primitive\_of(T3, M3) \wedge$
$(T2 \rightleftharpoons T3) \wedge$
$object\_type((O2, T2), M2) \wedge$
$object\_type((O3, T3), M3) \wedge$
$O2 \cong O3 \wedge$
$\Rightarrow$
$O1 \cong O3$

This rule states that **if** model primitive $T1$ (in model $M1$) is *compatible* with model primitive $T2$ (in model $M2$), and that model object $O1$ is *fully equivalent* to model object $O2$, and that model primitive $T2$ is *compatible* with model primitive $T3$, and that model object $O2$ is *fully equivalent* to model object $O3$, **then** model object $O1$ *must be fully equivalent* to model object $O3$. This is the transitivity between two *fully Equivalent* concepts (note that the compatibility between model primitives are not transitive). The *Transitivity of Full Equivalence* allows knowledge that is common and sharable among different models to be transfered and communicated between models. It also provides a basis for (automatic and semi-automatic) support for obtaining and maintaining consistency between models.

All of the above rules are generic and can be used to check any two model concepts that are captured in any two models, as long as the relationships described in the rules above are applicable in the models. We propose a systematic and incremental way of deploying the above rules using a *Domain-Model (DM)* as a mapping medium. The *Global Consistency* can be reached among all models by exhaustively determining *Pair-wise Consistency* between all models assuming that *Local Consistency* has been reached within each model. This is the three-tier framework which uses incremental efforts to allow models (each described in different modelling languages) to be gradually added to the "consistent set of mod-

[10] Again, the same attribute may also be given a different name in different models. We simplify this in the formula. Indeed, attributes may also only be "similar" and not "fully equivalent", but we leave this discussion for future work.

els" to achieve the *Global Consistency*. This framework is described in detail in [3].

The process of achieving *Global Consistency* is an iterative one. It sometimes requires a revisit of the model design phase as (new) information has been discovered and added to the model. Since every model needs to maintain consistency with every other model, in the worst scenario, the checking and updating activities may continue infinitely and become computationally intractable (even to achieve the local consistency of a model may not necessarily be computed in general). However, in our experience so far, such occasions rarely occur if the modelling languages have been chosen to be compatible with each other and the models have been carefully built. Typically, when an update does trigger a few other updates it does not trigger an infinite loop.

4 Discussion

We found that *Multi-Perspective Modelling* was a suitable approach to illustrate the essence of the domain of Air Operations, since it is able to capture the different aspects of the domain and allows the presentation and analysis of concerned model concepts that was required for the project.

An important *MPM* issue is to maintain the consistency and coherence among models. Since *Domain-Model* functions as a *light-weight ontology*, common knowledge between models can be captured and automatic error checking assistances can be provided through it. We presented a small set of fundamental consistency rules based on the provision of *Domain-Model*. They are, by no means, a comprehensive list, but provide a starting point for future work. In fact, the *MPM* activity still remains largely a labour-intensive task and automatic support for model building activities at the semantic level is much needed.

Although *MPM* is an excellent way to allow one to simplify a complex domain by allowing the modeller and reader to focus on only the concerned issues without overburden themselves with other irrelevant details. However, the task of understanding or analysing a full set of enterprise models, in which each model is a "simplified view" of the domain, is still fairly complicated. Support tools which provide clear illustration of the semantics and implications of the models are much needed to help understand and quality-prove these models.

5 Conclusion

The *Multi-Perspective Modelling* approach has been adapted to describe a complex domain, Air Operation. We found this approach suitable and often necessary when such a complicated domain must be captured and understood. Although the *MPM* approach is valuable in describing and prescribing the context and operations of an organisation, one important issue is to ensure the quality of the built models is high. We propose a framework which makes use of a light-weight ontology, a *Domain-Model*, as the underlying concept sharing mechanism to allow knowledge sharing, and obtaining and maintaining consistency across models which are described in different modelling languages. This work illustrates how formal methods may provide a foundation to support a framework that is independent of modelling language and application domain knowledge. As a result, it enhances the process of model quality-assurance.

6 Acknowledgement

I would like to show my appreciation to Dave Robertson[11] for his valuable comments and Albert Burger[12] for proof-reading the paper. I also would like to thank the referees of this paper from *IJCAI'01 KM and OM* workshop for their feedbacks. I appreciate efforts of members of the AOEM team[14], John Kingston[13], Larry Tonneson[14], Mike McNeil[15], Martin Brown[11], Jean MacMillan[16], Harry Gore[17] and Mike Pietrucha[18] who provided domain knowledge of Air Operations.

This work is partly supported under the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC)[5], which is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University. The EPSRC and the Universities comprising the AKT IRC are authorised to reproduce and distribute reprints for their purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either express or implied, of the EPSRC or any other member of the AKT IRC.

References

[1] Grady Booch, James Rumbaugh, and Ivar Jacobson, *The Unified Modelling Language User Guide*, Object Technology, Addison-Wesley, February 1999.

[2] Yun-Heh Chen-Burger, *Formal Support for an Informal Business Modelling Method*, Phd thesis, Artificial Intelligence, The University of Edinburgh, 2001.

[3] Yun-Heh Chen-Burger, 'A knowledge based multi-perspective framework for enterprise modelling', *To be appeared as a technical report at AI, University of Edinburgh.*, (February 2001).

[4] Yun-Heh Chen-Burger, David Robertson, and Jussi Stader, 'Formal support for an informal business modelling method', *The International Journal of Software Engineering and Knowledge Engineering*, (February 2000). World Scientific Publishing Co.

[5] AKT Consortium. http://www.aktors.org, October 2000. Interdisciplinary Research Collaborations (IRC),

[11] Artificial Intelligence, University of Edinburgh, UK.
[12] Computer Science, Heriot-Watt University, UK.
[13] AIAI, University of Edinburgh, UK.
[14] Zel Technologies, USA.
[15] BBN, San Diego, USA.
[16] Aptima, USA.
[17] Dynamic Research Corp., USA.
[18] SM&A, System Solutions Group, USA.

Advanced Knowledge Technologies (AKT) Project. Partners: University of Southampton, Aberdeen, Edinburgh, Sheffield and Open University, UK.

[6] Ramez Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems*, Addison-Wesley, 3rd edn., 2000.

[7] Hans-Erik Eriksson and Magnus Penker, *Business Modeling with UML: Business Patterns at Work*, John Wiley and Sons, 2000.

[8] M. S. Fox and M. Gruninger, 'Enterprise modelling', *AI Magazine, AAAI press.*, 109–121, (Fall, 1998).

[9] Ulrich Frank, 'Multi-perspective enterprise models as a conceptual foundation for knowledge management', *Proceedings of Hawaii International Conference on System Sciences, Honolulu*, (2000).

[10] Benjamin N. Grosof, 'Courteous logic programs: Prioritized conflict handling for rules', Research report rc 20836, IBM, T. J. Watson Research Center, UK., (December 1997). http://www.research.ibm.com/rules/papers.html.

[11] IBM, London, UK, *Business System Development Method, Introducing BSDM*, 2nd edn., May 1992.

[12] IBM, UK, *Business System Development Method: Business Mapping Part1: Entities*, 2nd edn., May 1992.

[13] J.E.Dobson, A. J. C. Blyth, J. Chudge, and M. R. Strens, 'The ordit approach to organisational requirements', *Requirements Engineering: Social and Technical Issues*, (1994). London, ed. Jirotka and J.A.Goguen, Academic Press.

[14] Defense Advanced Research Projects Agency (DARPA) Program Joint Force Air Component Commander. Air operation enterprise modelling project. http://www.darpa.mil/iso/jfacc/index.htm. 1999-2001.

[15] Richard Mayer, Christopher Menzel, Michael Painter, Paula Witte, Thomas Blinn, and Benjamin Perakath, *Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report*, Knowledge Based Systems Inc. (KBSI), September 1995. http://www.idef.com/overviews/idef3.htm.

[16] National Institute of Standards and Technology, *Integration Definition for Function Modelling (IDEF0)*, December 1993.

[17] Martyn A. Ould, *Business Processes: Modelling and Analysis for Re-engineering and Improvement*, John Wiley and Sons, 1995.

[18] G. Schreiber, B. Wielinga, and J. Breuker, *KADS: A Principled Approach to Knowledge-Based System Development*, Academic Press, November 1997.

[19] Bernhard Thalheim, 'An overview on semantical constraints for database models', *Proceedings of The 6th International Conference on Intellectual Systems and Computer Science, Moscow, Russia*, (December 1996).

[20] Pepijn R. S. Visser, T.J.M. Bench-Capon Dean M. Jones, and M.J.R. Shave, 'Assessing heterogeneity by classifying ontology mismatches', *Formal Ontology in Information Systems, Proceedings of FOIS, IOS Press, Amsterdam, The Netherlands.*, 148–162, (1998).

[21] John A. Zachman. The framework for enterprise architecture. http://www.zifa.com/ and http://www.barnettdata.com/fromzifa.htm. Zachman Institute for Framework Advancement.