Chapter 16

SHARING AND CHECKING ORGANISATION KNOWLEDGE

Yun-Heh Chen-Burger
Artificial Intelligence Application Institute, The University of Edinburgh,
80 South Bridge, Room E32, Edinburgh EH1 1HN, UK
http://www.aiai.ed.ac.uk
jessicac@aiai.ed.ac.uk

Abstract The approach of *Multi-Perspective Enterprise Modelling* is now more commonly accepted and used in practice as a way to manage organisational 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, such organisational knowledge, but that much automated support is needed.[1]

Keywords: Knowledge Management, Ontology, Multi-Perspective Modelling, Business Model, BSDM, Enterprise Model, Process Model, Knowledge-Based Support Tool, Knowledge Sharing, 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.

16.1 INTRODUCTION

The *Multi-Perspective Modelling (MPM)* technique involves using multiple modelling languages to describe a domain from different points of view, which allows relevant knowledge to be presented in a concise and focused manner. The *MPM* approach is useful and sometimes necessary to capture *corporate knowledge* because organisational knowledge is often so complicated and of heterogeneous types that no single modelling method can capture all of the important aspects and present them clearly and appropriately.

The *MPM* approach is used in research and practice: Common KADS methodology (Schreiber et al., 1997) embodies several modelling languages to help understand and capture domain knowledge and to help the design of knowledge based systems; Booch, Rumbaugh and Jacobson (Booch et al., 1999) embrace this approach and offer a suite of inter-supportive modelling notations in the *Unified Modelling Language*; Frank(Frank, 2000) uses this approach to design and build a multi-perspective knowledge management system (MEMO); Zachman(Zachman, 1987) uses a variety of modelling languages to capture and describe the different aspects of a domain. The importance and benefits of using multiple modelling languages to represent a complex knowledge body is well-recognised and the MPM approach has been adopted by many.

During the *Air Operation Enterprise Modelling* project(AOEM, 1999), 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[2]. 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). The above information is aided with informal documents, workshop and email correspondence with domain experts.

Several aspects are considered: the infrastructures used during the operations, the operations that are carried out, people involved and their actions, policies that are followed, resources and information needed, and issues such as timing for cooperation during the operation. To illustrate these aspects, three types of models are built: a *Domain-Model* to provide a taxonomic structure to capture all the high-level and fundamental concepts, a *Business Model* to capture the infrastructure

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

and the detailed concepts involved in the operations, and a *Role Activity and Communication Model* to identify the type of actors who are involved in the operations, their operations and the interactions between them.

Although these models were appropriate to the needs of the project, we constantly faced the problem, as any *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 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 our modelling efforts carried out under the *AOEM* project and the initial work on automatic consistency checking on multiple enterprise models under the *IRC AKT* project(AKT, 2000).

16.2 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 chosen modelling languages must be suitable for the problem domain and appropriate to achieve the modelling objectives. Secondly, the chosen modelling languages should be complementary of each other and all concerned knowledge can be described using 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 these models are described at a similar level of abstraction: if some models allow multiple levels of abstraction, appropriate guidelines must be established to determine which level of abstraction is mapped to other models.

Figure 16.1 shows our *MPM* approach. As mentioned earlier, three models are used: *IBM BSDM's Business Model (BM)*(IBM, 1992), *Domain Model (DM)*(Chen-Burger, 2001a), and *Role Activity and Communication Diagram (RACD)*(AOEM, 1999).[3] 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 more than one model, each using its specialised modelling primitives.

[3] *RACD* notation was developed by the author to meet requirements for AOEM project. It was adapted and extended from *Role Activity Diagram*(Ould, 1995) and *IDEF3*.

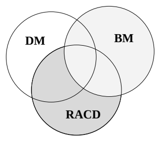

Figure 16.1. Overview of Multi-Perspective Modelling Approach

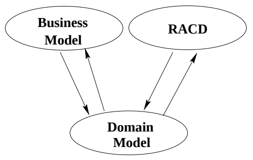

Figure 16.2. MPM using Domain-Model as a Backbone

The area that is covered by only one model denotes the specialisation of the particular modelling language that describes the type of knowledge that is not (or can not be) captured by any other models. The Domain Model may cover some or all of the overlap between BM and RACD. Figure 16.2 depicts how *Domain-Model (DM)* serves as a *light-weight ontology* in the *MPM* approach. It provides a taxonomic structure to store the fundamental and important knowledge of the domain. Two types of knowledge are captured: the high-level classification information about the domain and the (lower-level) model concepts that are 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.

Because the information stored in the *DM* is common and sharable between different models, it is a natural media for knowledge transfer and translation. Model concepts that are described in one model are mapped to *DM* which are then mapped to another model. Given an appropriate mapping mechanism, knowledge can be matched, shared and translated between models. This mapping mechanism has been implemented in a rule-based system, *KBST-EM*(Chen-Burger, 2001b). Note that although Domain-Model explicitly represents the *is-a* relationship, it does not imply an Object-Oriented paradigm. If an Entity-Relational view of the knowledge is desirable, an ER data model may be included in the enterprise models.

16.3 OBTAINING AND MAINTAINING CONSISTENCY

Visser(Visser et al., 1998) identifies four categories of heterogeneity existing between bodies of information: i.e. *paradigm, language, ontology* and *content heterogeneity.* This paper focuses on dealing with models that have *language* and *content heterogeneity*, since those models are written in different modelling languages (language heterogeneity) and may describe different parts and aspects of the same domain (content heterogeneity).

A set of 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 lies within the modeller's control.

We use A $\cong$ B to denote that model concept A is (conceptually) *fully equivalent* to model concept B, i.e. A and B are mapped to the same concept in the Domain Model. We use A $\risingdotseq$ B to denote that model primitive A is *compatible* with model primitive B, where A and B may be used in different models. To judge whether model primitives A and B are compatible, they must represent a similar function in their own modelling languages. Two types of inference operators of different strength of enforcement have been deployed: A $\Rightarrow$ B indicates if A is true then B *must be* true, A $\triangleright$ B indicates if A is true then B *may be* true. The set of consistency rules is given below.

(1) Consistent Representation of Information

$\forall T1, M1, T2, M2, O1, O2, Att, Value1, Value2.$

$model\_primitive\_of(T1, M1) \wedge model\_primitive\_of(T2, M2) \wedge$

$object\_type((O1, T1), M1) \wedge object\_type((O2, T2), M2) \wedge$

$(T1 \risingdotseq T2) \wedge (O1 \cong O2) \wedge$

$object\_attribute\_in\_model((Value1, Att), (O1, T1), M1) \wedge$

$object\_attribute\_in\_model((Value2, Att), (O2, T2), M2)$

$\Rightarrow Value1 = Value2$

where *model_primitive_of(T1, M1)* indicates $T1$ is a model primitive (type) of model $M1$; *object_attribute_in_model((Value1, Att), (O1, T1), M1)* stores the attribute value $Value1$ for attribute *Att* for model object $O1$ in model $M1$; *object_type((O1, T1), M1)* defines the model object $O1$ is of model primitive type $T1$ in Model $M1$. This formula indicates **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.[4] This consistency rule ensures information that is shared across models is consistently repre-

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

sented in all models.

(2) Correct Specialisation of Concepts

$\forall T1, M1, T2, M2, O1, O2, S1.$
$model\_primitive\_of(T1, M1) \wedge model\_primitive\_of(T2, M2) \wedge$
$object\_type((O1, T1), M1) \wedge object\_type((O2, T2), M2) \wedge$
$T1 \risingdotseq T2 \wedge O1 \cong O2 \wedge object\_type((S1, T1), M1) \wedge sub\_type(S1, O1, M1)$
$\Rightarrow$
$\neg \exists S2. object\_type((S2, T2), M2) \wedge S1 \cong S2 \wedge sub\_type(O2, S2, M2)$

where $sub\_type(S, O, M)$ denotes that model concept S is a sub-type or specialisation of model concept O in model M. This formula imposes a consistent definition on the specialisation of concepts across all models, i.e. **if** model objects, $O1$ and $O2$, in model $M1$ and $M2$, are *fully equivalent*, and $S1$ is a sub-type of $O1$, **then** it *must not* be the case that another concept $S2$ is found in model $M2$ that is fully equivalent to $S1$ and is the super-type 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.

(3) Consistent Application of Dependencies

$\forall T1, M1, T2, M2, O1, D1.$
$model\_primitive\_of(T1, M1) \wedge model\_primitive\_of(T2, M2) \wedge$
$object\_type((O1, T1), M1) \wedge object\_type((O2, T2), M2) \wedge$
$T1 \risingdotseq T2 \wedge O1 \cong O2 \wedge object\_type((D1, T1), M1) \wedge depends\_on(O1, D1, M1)$
$\triangleright$
$\neg \exists D2. object\_type((D2, T2), M2) \wedge D1 \cong D2 \wedge depends\_on(D2, O2, M2)$

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

Figure 16.3 gives examples of where dependencies may be derived and the above rule may be applicable. Three models are given: a process model, a data model and a business model. Figure (a) features a commonly seen structure in a process model[5] where the execution of process

[5]e.g. *IDEF0(IDEF0, 1993), IDEF3(Mayer et al., 1995), and RACD(Chen-Burger, 2001b)* models.

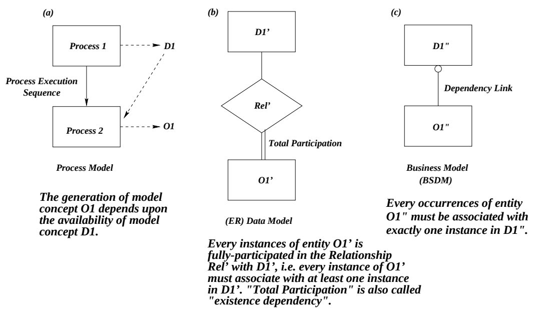
(a)
Process 1
D1
Process Execution Sequence
Process 2
O1
Process Model
The generation of model concept O1 depends upon the availability of model concept D1.
(b)
D1'
Rel'
Total Participation
O1'
(ER) Data Model
Every instances of entity O1' is fully-participated in the Relationship Rel' with D1', i.e. every instance of O1' must associate with at least one instance in D1'. "Total Participation" is also called "existence dependency".
(c)
D1"
Dependency Link
O1"
Business Model (BSDM)
Every occurrences of entity O1" must be associated with exactly one instance in D1".

Figure 16.3. Example Dependencies in Different Models

2 is preceded by process 1 which produces information $D1$ that is used to generate information $O1$ by process 2. Based on which, one can derive model object $O1$ depends on model object $D1$, since instances of $O1$ can not be created unless the corresponding instances of $D1$ have been created. Figure (b) shows a data model, an *Entity-Relational* data model in this case. A special type of relationship, *Total Participation*, has been used to indicate an *Existence Dependency* (Elmasri and Navathe, 2000) which constrains that every instance of entity $O1'$ must be related to at least one instance of entity $D1'$ in the relationship Rel'. Figure (c) specifies model object $O1''$ depends on $D1''$ in a similar one-to-many dependency link in a business model.[6]

The above models suggest dependency is a common property in many different modelling languages and it can be extracted and generalised. The dependency described in the rule above is generic, i.e. it may include "dependencies" of different strength in different models. The weak inference operator $\triangleright$ is therefore used to include all cases and allow modelling flexibility. When specific dependencies are used, the strong inference operator, $\Rightarrow$ (must be), can be used.

(4) Detecting Incompleteness

$\forall R1, M1, R2, M2, T1, T2, O1, P1, O2, P2.$
$relationship\_type(R1, M1) \wedge relationship\_type(R2, M2) \wedge$
$model\_primitive\_of(T1, M1) \wedge model\_primitive\_of(T2, M2) \wedge$
$R1 \risingdotseq R2 \wedge T1 \risingdotseq T2 \wedge$
$object\_type((O1, T1), M1) \wedge object\_type((O2, T2), M2) \wedge$
$object\_type((P1, T1), M1) \wedge object\_type((P2, T2), M2) \wedge$

[6]This notation is taken from *IBM's Business Model in BSDM(IBM, 1992).*

$O1 \cong O2 \wedge P1 \cong P2 \wedge in\_relation(R1, O1, P1, M1)$
$\triangleright\, in\_relation(R2, O2, P2, M2)$
where the predicate $in\_relation(R1, O1, P1, M1)$ specifies model object $O1$ is associated with $P1$ in the relationship $R1$ in model $M1$. Given 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 relationship, the above rule is generic and a weak inference $\triangleright$ is used. The rule represents an example completeness analysis as a part of the consistency checking process. Similar principles may be applied to detect specific missing information and specific results may be suggested.

(5) Inferring Missing Information

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

This formula is a weaker version of consistency rule (1). It indicates **if** two model objects, $O1$ and $O2$, are *fully equivalent*, and object $O1$ has an attribute Att with value $Value$, **then** object $O2$ in model $M2$ *may* also have the same attribute Att with value $Value$. This rule suggests information that is described in one model may be usefully described in another model. However, since each model is meant to serve different aims, it is not necessary that such information is always duplicated. This avoids flooding a model with excessive information, hence a weaker inference symbol has been used, $\triangleright$.

(6) Transitivity of Full Equivalence

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

This rule indicates **if** model object $O1$ is *fully equivalent* to model object $O2$, and model object $O2$ is *fully equivalent* to model object $O3$, **then** $O1$ *must be fully equivalent* to $O3$, given their corresponding model primitives are compatible. This is the transitivity of *Full Equivalence*. It 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 may be used to check model concepts in different models when applicable. We propose a systematic and incremental way of deploying the above rules in a *Three-tier Framework* (Chen-Burger, 2001b). The *Global Consistency* can be reached among all models by exhaustively achieving the *Pair-wise Consistency* between all models and *Local Consistency* within each model. The process of obtaining *Global Consistency* is iterative. It may require a revisit of the model design phase as (new) information has been discovered and added to the model. Theoretically, in the worst scenario, the checking and updating activities may be infinite. However, in our experience such occasions rarely occur if modelling languages have been chosen to be compatible with each other and models have been carefully built. Typically, when an update does trigger a few other updates it does not trigger an infinite loop.

16.4 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.

Acknowledgement

The author would like to thank Dr. Dave Robertson (Artificial Intelligence, University of Edinburgh, UK) for his valuable comments, Dr. Albert Burger (Computer Science, Heriot-Watt University, UK) for proof-reading the paper, and comments from referees of *IJCAI'01 KM and OM* workshop. The author appreciates efforts from AOEM team(AOEM, 1999), John Kingston(AIAI, University of Edinburgh, UK), Larry Tonneson (Zel Technologies, USA), Mike McNeil and Martin Brown (BBN, San Diego, USA), Jean MacMillan (Aptima, USA), Harry Gore (Dynamic Research Corp., USA) and Mike Pietrucha (SM&A, System Solutions Group, USA) who provided domain knowledge of Air Operations. This work is partly supported by AKT IRC Project. AKT is sponsored by the UK EPSRC under grant number GR/N15764/01. 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

AKT (2000). Advanced knowledge technologies project. interdisciplinary research collaboration, uk. http://www.aktors.org.

AOEM (1999). Air operation enterprise modelling project, darpa program. http://www.darpa.mil/.

Booch, G., Rumbaugh, J., and Jacobson, I. (1999). *The Unified Modelling Language User Guide*. Object Technology. Addison-Wesley.

Chen-Burger, Y.-H. (2001a). *Formal Support for an Informal Business Modelling Method*. Phd thesis, Artificial Intelligence, The University of Edinburgh.

Chen-Burger, Y.-H. (2001b). A knowledge based multi-perspective framework for enterprise modelling. *Technical report at AI, University of Edinburgh.*

Elmasri, R. and Navathe, S. B. (2000). *Fundamentals of Database Systems*. Addison-Wesley, 3rd edition.

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

IBM (1992). *Business System Development Method, Introducing BSDM*. IBM, London, UK, 2nd edition.

IDEF0 (1993). *Integration Definition for Function Modelling (IDEF0)*. National Institute of Standards and Technology.

Mayer, R., Menzel, C., Painter, M., Witte, P., Blinn, T., and Perakath, B. (1995). *IDEF3 Process Description Capture Method Report*. Knowledge Based Systems Inc. (KBSI). http: //www.idef.com.

Ould, M. A. (1995). *Business Processes: Modelling and Analysis for Reengineering and Improvement*. John Wiley and Sons.

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

Visser, P. R. S., Dean M. Jones, T. B.-C., and Shave, M. (1998). Assessing heterogeneity by classifying ontology mismatches. *Formal Ontology in Information Systems, Proceedings of FOIS, IOS Press, Amsterdam, The Netherlands.*, pages 148–162.

Zachman, J. A. (1987). A framework for information systems architecture. *IBM Systems Journal, IBM Publication.*, 26(3). Relevant web site: http://www.zifa.com/.