Modeling Technology for a Model-Intensive Enterprise

Peter Denno, David Flater, Michael Gruninger

***Abstract*—The Object Management Group (OMG™) is defining specifications supporting a modeling environment that permits models from a 'family of modeling languages' to populate a repository. In that environment, mechanisms that give coherence to the collection of models defining a project are the repository's common meta-model and constraints enforcing a notion of well-formedness on the content of the repository. This paper argues that additionally what is required to implement a model driven architecture (MDA™) is an 'inter-model' language with reach beyond an object context into an ontology of the domain vocabulary and requirements definition. If specifications under this arrangement are modularized properly, reuse of the domain terminology and business sector characterization is possible. The repository federates models into an emergent enterprise model. This paper considers opportunities for an inter-model language in supporting traceability of requirements from domain knowledge to implementations. It explores an idea of modularized specifications where domain knowledge is separated from implementation-biased models and reified refinement relations bridge the gap.**

***Index Terms*—enterprise modeling, meta-data repositories, ontologies, requirements engineering, traceability.**

I. Introduction

This paper explores how a model repository based on a family of modeling languages and an inter-model language (IML) may be used to federate project-oriented models into an emergent enterprise model. The benefits that federation might yield are a non-prescriptive approach to enterprise modeling, a reusable domain characterization, traceability, and easier system validation.

Peter Denno (peter.denno@nist.gov) and David Flater (david.flater@nist.gov) are with the National Institute of Standards and Technology, Gaithersburg, Maryland 20899 USA.

Michael Gruninger (michael.gruninger@nist.gov) is with the University of Maryland, College Park, Maryland, USA.

Among the myriad problems encountered in large integration projects, we focus on problems of *coherence among models,* of which the following are examples: (1) current development practices seldom make use of an enduring characterization of the environment[1]. Subsequent business process re-engineering repeats the costly work of characterizing the environment; (2) typically, modeling that might allow domain experts to verbalize and verify domain knowledge cannot be materially used in subsequent implementation-biased modeling, and; (3) the differing perspectives and representations implied by the various modeling technologies employed – though essential to characterizing the system – hinder an account of the refinement of thought embodied in the models. This undermines traceability.

These problems are well known and are the subject of rich bodies of research. Further, they are interrelated; they each concern the proper separation and interrelationship of information embodied in many models defined from disparate viewpoints. The problems need not be addressed independently. The paper explores the idea that solutions can be realized simultaneously using a model repository having an integral *inter-model language* (IML) implemented as a description logic. Within the model repository, the IML would provide two functions: (1) it would provide a means to establish refinement relationships between elements of project-oriented models; and, (2) it would provide a means to relate the project-oriented models to its own conceptualization of the environment. The language and repository would, in effect, allow an enterprise model to be defined 'bottom-up' as the enterprise's various information infrastructure and business process re-engineering projects provide models to be federated into a coherent whole. Though the approach does not prescribe an enterprise model, it may require rigorous attention in the areas of conceptual modeling and requirements engineering.

This paper describes very early work toward these goals. The problems are discussed in the context of a meta-data repository and *family of languages* [1],[2] concepts as promulgated by the Model Driven Architecture™ (MDA™) [26], [23], [27] of the OMG. The model federation approach discussed here would provide an MDA with *traceability* – an account of the relationship between motivating statements of

[1] - *environment* refers to the context under which a developing project will operate, including business rules and processes, organizational structure, and existing information technology. A characterization of the environment is unbiased with respect to the implementation of developing projects. This usage is similar to that of [3].

requirements and their realization in system behaviors.

The next section of the paper describes how a family-of-languages, meta-data repository and reified refinements may provide coherence among models. *Section III* considers how a description of the environment and requirements should be organized to ensure reusability. The idea that environment information and refinements may be represented using an IML is discussed. *Section IV* considers how knowledge about the environment should be modularized to make the best use of existing sources and ensure its coherence and stability. *Section V* reviews related work. *Section VI* concludes the paper with a discussion of possible future work.

II. Model Repository and Refinement

A. Requirements

In the course of a typical project, various modeling technologies, possessing differing viewpoints, levels of abstraction, fidelity, and formality are employed. Software engineering methodologies differ principally with respect to the path of refinements they prescribe. Disregarding these differences, the succession of models proceeds, roughly speaking, from an elaboration characterizing the operating environment and requirements to an elaboration of implementations.

A facility that may foster coherence among project-oriented models in this broadly defined scenario should provide the following: (1) a means to manage the collection of project-oriented models (a model repository); (2) a means to define the environment under which the project-oriented models are meaningful; and, (3) a means to relate the project-oriented models to each other and to the environment. These requirements underlie the design choices described below. The model repository is considered first.

B. Model repository

A model repository may serve to increase coherence across models by various means, among which are concern spaces [4], commonality in usage of meta-model elements across the various modeling technologies, and explicit placement of refinement relationships. Concern spaces, though promising, are not discussed further here. Commonality in usage of meta-model elements is pursued in the OMG through the development of the Meta Object Facility™ (MOF™) [30] and family-of-languages notion for UML™ .

UML [6], [28] is defined using a subset of itself called the *UML meta-model*. The meta-model defines class definitions (*e.g.* a class representing UML associations, a class representing UML classes *etc.*) and describes the intent of those classes with English language text. Well-formedness rules written in the Object Constraint Language (OCL) [29],[15] constrain the usage and interrelation of instances of those classes (that is, a model written in UML) to meaningful forms. The family-of-languages approach provides a methodical means to define modeling notions that differ in form and meaning from those that UML defines.

The family-of-languages of UML is achieved by either of two means: UML profiles or the OMG Meta-Object Facility (MOF). The profile approach uses capabilities within UML (stereotypes, tagged values and constraints) that, for all practical purposes, allow one to subclass a UML model element for one's particular purposes. The MOF approach is more far-reaching; it is based on the fact that the UML meta-model is itself an instance of a model, the MOF model. By instantiating additional objects from the MOF model (new model elements in the UML meta-model) and by defining associations between these elements and the model elements of the UML meta-model, new modeling technology may be defined.

The MOF-based approach is more comprehensive than the profile-based approach in that it may be used to instantiate entirely new classes, not just subclasses of UML meta-model classes. The perspective on a model repository advanced by this paper favors the MOF-based approach, in part because it is likely that there will always be useful modeling notations that are not simple specializations of UML.

The MOF is an instance of a layered architecture for a meta-data repository. (See *Figure 1*.) An earlier presentation of this idea is the Information Resource Dictionary System (ISO 10027) [5].

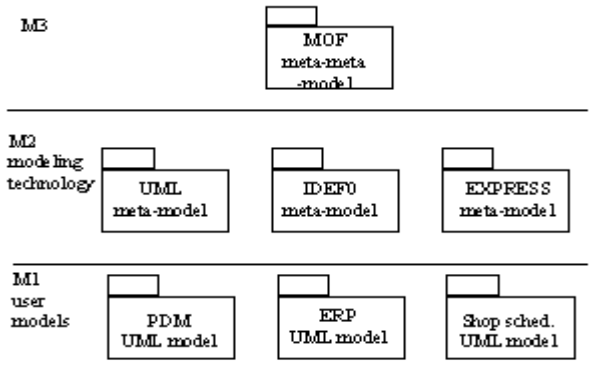

Figure 1 – Layered meta-data architecture

It is of great advantage for the modeling technology of project-oriented models to share common modeling notions through a layered meta-model architecture. A meta-model that instantiates the modeling technologies of a collection of domain models provides elements of a common language for making statements about those modeling technologies and the models they instantiate.

C. Reified refinements and concerns

A second means to foster coherence among models of a project is to identify it explicitly.

A *refinement* is a relationship between model elements that represents the elaboration of details existing between the elements. *Model elements* are instances representing pieces of a model and are instantiated from the meta-model of the

model. (All the various meta-model objects: classes, associations, packages, are derived from a model element class.)

The refinements made in the course of a project may span models and documents defined from various viewpoints. *Traceability* is an account of the relationship between motivating statements of requirement and their realization in system behaviors. From the perspective of this paper, traceability is supported as an account of refinements between model elements.

Though UML possesses a notion of refinement, it is rather inchoate; it "specifies refinement relationship between model elements at different semantic levels, such as analysis and design" [6], *pg. 2-19*. A direction for this research is to find a firmer foundation and elaboration of the notion.

Refinements should be reified objects. The value of treating refinements as first class objects in their own right is the rigor that this brings to the notion of traceability: if refinement relationships are reified and classifiable, and if the model elements to which they refer are identified, then accounting for system behavior (and making domain experts and developers accountable for specific characteristics of the system) can be more easily systemized. These are very big 'ifs', however.

A classification of refinements will add to their utility. Following roughly the reasoning of [7], three dimensions of refinement may be identified: (1) *conceptual refinement*, where the model elements are terms from the universe of discourse, and the refinement is a constraint concerning their usage; (2) *behavioral refinement*, where the model elements represent actors and the relationship concerns the representation of synchronization, triggering, conditions, composition or factoring of activity; and, (3) *technical refinement*, a relationship in which an implementation commitment is made. A specialization of technical refinement is *structural refinement*, where the model elements describe an encoding of information, and the refinement concerns differences in these encoding. In addition to these refinement relationships, there may be *identity* and *conflicting assertion* relationships between model elements. (See *Figure 2*).

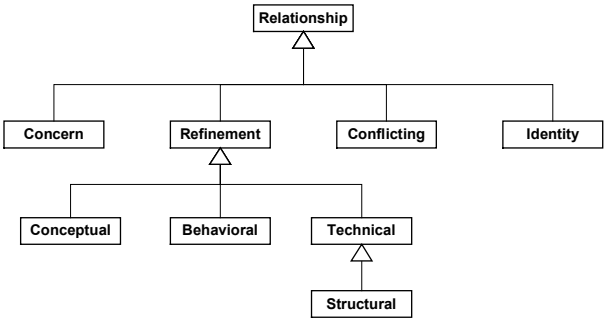

Figure 2 — Classification of relationships among model elements

Technical refinement occurs only between model elements of implementation-biased models. Behavioral refinements occur in either conceptual or implementation-biased models.

Refinements are distinguished from concerns. A *concern* is a relationship between a statement about the environment (which is necessarily conceptual) and its manifestation in (implementation-biased) model elements.

Conceptual refinement should occur only in conceptual models (models lacking an implementation bias). In the architecture described, there would be no conceptual refinement relationships among the model elements in the repository (project-oriented models). Such refinement would reflect new concepts introduced through implementation and unrelated or unknown in the operating environment. Existence of such a refinement hinders reusability of knowledge of the environment. The relationship between a conceptual assertion and its manifestation in model elements is a *conceptual concern*. These ideas are discussed further in section III.

Clearly, refinements and concerns encompass a wide body of information, the foundation of knowledge from which they emerge is immense. Though the characterization above may be useful towards the goals of coherence, they are not the sort of distinction typically made by software developers; that is, they concern the nature of the refinement, not the domain nor its models. Identification and placement of refinements requires additional modeling effort.

Identification of refinements can be aided by mechanisms that exploit an understanding of the modeling technologies involved. In the context of a four-layer architecture, where modeling technology resides at the M2 layer, refinement relationships occur at the M1 level (*i.e.* in user models), an 'image' of the refinement might occur at the M2 level. That is, between modeling technologies themselves a justification for M2 refinements may exist. For example, the IDEF0 [8] notion of control does not distinguish pre-conditions from triggering events, whereas UML state machines, [6] and PSL [9] do. *Figure* 3 illustrates the idea, depicting one (of potentially many) refinement images between IDEF0 and UML state charts.

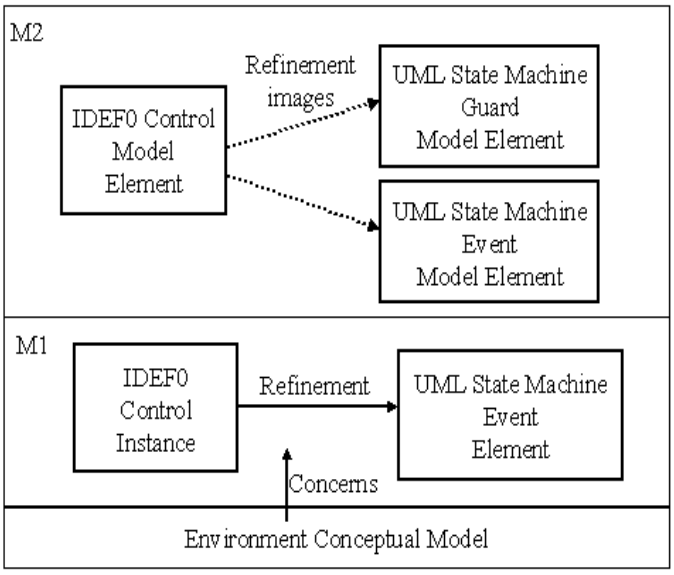

Figure 3: — A behavioral refinement and its image

III. Modeling the environment and requirements

A model repository based on a layered meta-model architecture and reified refinements may provide coherence among project-oriented models. Traceability to requirements requires the establishment of additional relationships, those from elements of project-oriented models to the operating context and statements of requirement. If careful attention is paid to the separation of implementation concerns from environment concerns, conceptual and enterprise modeling effort can be effectively used in subsequent refinement and reused in subsequent business process re-engineering. These ideas are considered below.

A. Separation from implementation dependencies

Developers of information technology typically interview domain experts to acquire knowledge of business processes and terminology. This knowledge may be recorded in lexicons, use case models and various other conceptual models. Without an enduring infrastructure for its integration with other models, refinement relations cannot be established between this body of knowledge and subsequent models. Traceability is lost. Without separation from implementation concerns, the portion of knowledge containing an implementation bias is often proven through refinement of the implementation to be incorrect. In practice, conceptual models are typically treated as by-products of implementation, useful for verifying facts with domain experts and helping developers to gain familiarity with the domain of discourse, but providing little additional and subsequent utility.

A number of researchers in requirement engineering and conceptual modeling have emphasized the importance of keeping characterizations of requirements and environment free of implementation bias [10], [3], [11], [32]. A characterization of the environment that is free of implementation detail is more easily verbalized and verified by domain experts. Further, its potential for reuse is greatly improved.

In [3], Zave and Jackson argue for additional distinctions, specifically between (1) a *designation* (informal description of the meaning of an atomic formal term referring to the environment) and a *definition* (a formal term built ultimately on designations); (2) actions initiated by the environment, actions initiated by the system and actions shared by the system and environment, and; (3) environment state and system state. Regarding the latter, they argue that characterizations of system state introduce an implementation bias. To avoid this, assertions about system state can be reformulated as characterizations of the environment with and without the system. The advantages that this discipline affords, they argue are: (1) the ability to collect information about the environment without regard to how it fits within the system to be constructed; (2) a distinction that highlights the potential discrepancy between the environment state and system state (this might serve validation purposes), and; (3) a stable relationship between requirements, specifications and domain knowledge.

B. Towards an inter-model language

In the light of the above, the operating environment including definitions, business rules and processes may be established in an IML. This information constitutes aspects of an enterprise model, invariant to implementation concerns that do not affect business process. In order to achieve coherence to requirements, relationships may be established between this information and project-oriented models. These ideas are discussed below.

Refinement relationships may occur between model elements of disparate models. The relationship between refinements and the models is analogous to the relationship between Object Constraint Language (OCL) and elements within a single UML model – both 'reside outside' the elements they reference and establish a context based on those elements.

A model of the environment that is reusable with respect to various implementation specific models will have separate sets of refinements to those models arising from the requirements of the various projects. In this sense, the environment model also resides outside the implementation specific models.

The observations of the previous two paragraphs provide some justification for the use of an *inter-model language*. In order to address uniformly the requirements engineering concerns, this language must have reach beyond the object context (*i.e.* beyond the context of OCL) into an ontology of the domain terminology and characterization of the environment. In summary of the considerations of *Section III A*, above, the following additional requirements are apparent: (1) it should allow specification of terminology, with clear indication of what is a designation; (2) it should be possible to seamlessly extend the language to express notions of time and process; (3) it should be possible to distinguish actions that are performed by the environment from actions that are performed by the system or shared, and; (4) it should possess the qualities of a good conceptual modeling language, (such as enumerated by [10]) expressability, clarity, semantic stability, validation mechanism, and formal foundation.

More complete arguments than made here for (1) and (3) can be found in [3]. The need to describe actions in the environment necessitates (2). Description logic such as Powerloom [12] and KL-ONE [13] as well as knowledge management and acquisition tools such as CODE4 [14] may generally fill these requirements. A process specification language such as PSL [9] may serve (2). A detailed discussion of these is beyond the scope of this paper.

C. An example

The following example illustrates the idea of separation of conceptual modeling into assertions in the IML and linking of assertions to a project-oriented (implementation-biased) model. The example is based on one provided by Warmer and Kleppe [15]. The figure (*Figure 4*) depicts a UML class diagram from a finance domain. The model allows that a person have a mortgage that takes as security a house that is owned by another person. As the authors point out, this is not

realistic and OCL may be written to prohibit that interpretation.

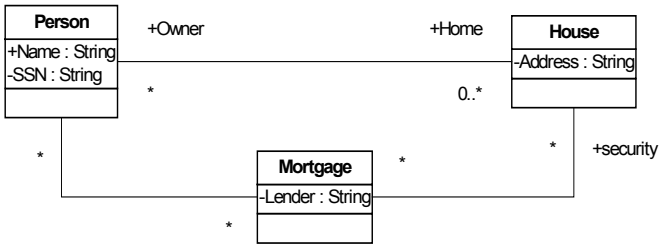

Figure 4 — UML Class diagram for mortgage

This model, contained in the model repository, is an instantiation of M2 classes of the UML meta-model. Specifically, instances of UML:Class objects for *Person*, *House* and *Mortgage* and instances of UML:Association objects connecting the objects. The UML:Class and UML:Association classes are themselves instances of M3 level classes

In predicate logic (representing what might be IML) the constraint could be written as:

∀ (Person P)(House H)(Mortgage M):
(owns P H) ←(has-mortgage-for P M H)

The concepts *Person*, *House*, and *Mortgage* defined in the IML may be either designations (*i.e.* described informally) or definitions, (*i.e*. defined by expressions built ultimately on designations) the choice depending on one's modeling preferences. *Person*, *House* and *Mortgage* are related to the corresponding classes of the UML model through *technical concern* relations. (The UML model commits to a specific object-oriented implementation). The association *owner* between *Person* and *House* represents an application of the concept *owns* (a definition) defined originally in the IML. Concepts should not be introduced nor refined in implementation models. The relationship between *owns* and the OCL constraint is a *conceptual concern*. This is depicted in *Figure 5*, below.

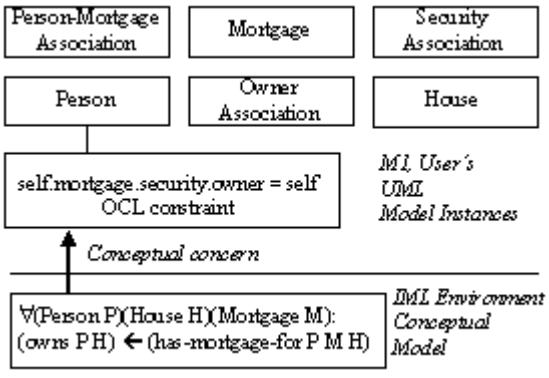

Figure 5 — Example conceptual concern

D. Advantages of separation

Constraints defining a purely conceptual model may be collected without regard for particular implementations. When described in a logical formalism, a collection of such knowledge may be tested for logical consistency. Further, though it may be argued that the rule above cannot be verbalized easily by an expert, it is probably significantly more easily verbalized than the same constraint embodied in an implementation-biased UML model. Finally, ordinary objective statements of a conceptual model may be presented in notations such as Object Role Modeling, [10], [16] that some domain experts may find to be more easily verbalized than quantified logical statements. Promising approaches to aiding experts in articulating domain knowledge are described by Lethbridge [14] and Sharp [17].

Finally, as the example may illustrate, accountability originates in the environment, but not necessarily through requirements analysis with a domain expert. It can be argued that the mortgage security constraint is not really 'discovered' in an interview with the domain expert but rather constitutes commonly held belief. If this seems unlikely at a level of specificity concerning mortgages and security, it seems more likely concerning knowledge of loans in general. 'Discovering' such knowledge again and again in interviews is wasteful.

In a description logic model of this domain, a *House* may be a kind of *Valuable-thing* and a *Mortgage* may be kind of *Secured-loan*. In that context, a more general version of the example rule might apply:

∀(Person P) (Valuable-thing V1) (Secured-loan L)
∃(Valuable-thing V2):
(AND (has-secured-loan-for P L V1)
(loan-security L V2)) ➔ (OR (owns P V2) (= V1 V2))

This latter rule might apply as well to a home equity loan to pay college tuition.

In summary, a formal inter-model language aids consistency checking of the knowledge asserted by domain experts and is more easily verbalized than models obfuscated by implementation concerns. If the formal language is a description logic, the further benefits of default reasoning in a manner paralleling a programming language's type hierarchy, and sophisticated query capability might be realized.

As noted above, the identification of refinements, and rigorous attention to keeping characterizations of the environment and requirements free of implementation-bias, exacts a price on the development process. The next section of the paper explores how some of the potentially costly work involved can be avoided through the reliance on information found in consensus industrial standards.

IV. An Emergent Enterprise Model

An enterprise model is a model that serves to inform regarding the organizational structure, resources, information technology and processes of an enterprise. The model is of

crucial value in business process re-engineering efforts.

Use of the architecture described entails encoding many of the concerns of enterprise modeling into a conceptual model defined in IML. These concerns include descriptions of business process with and without the information technology produced by the project, and definitions of fundamental terms and business rules. Further, this information is linked to its implementation in project-oriented models. In this sense, the architecture enables the specification of an enterprise model 'bottom up'.

There are two issues to discuss with respect to the architecture and enterprise modeling. The first is how the enterprise might establish the foundational concepts of its enterprise model. The following suggests that consensus industrial standards may help here. The second issue is how the information might be organized into enterprise model viewpoints.

An important precondition for the development of a consensus industrial standard is the acceptance among participants of a common usage of terms, and of a common understanding of the business domain for which the standard encodes best practice. These requirements are at the foundation of standards making. Consensus building towards these requirements usually relies on prior studies. In the area of manufacturing system integration standards (where the authors work) for example, there is a large body of research characterizing manufacturing domains and the environment in which information technology for manufacturing operates. Such work may serve as the foundation of knowledge for a usage of the architecture. That is, a conceptual expression of requirements in the form described *in Section III* can be built from a characterization of the business domain, and terminology. The latter are established by consensus industrial standards.

Further, standards often specify archetypal use cases and corresponding conformance classes for uses of portions of the specification. Though the idea is only distant on the horizon, consensus industrial standards that build their terminology on the foundation of a standard upper ontology [18] might integrate well with the architecture described here. In this scenario, the standard may formally describe terminology and use cases for normatively defined interfaces. An archetypal process description corresponding to a use case of the interface (sometimes called a *recommended practice*) may be specialized by an enterprise's description of the usage. This 'layering of description' is illustrated in *Figure 6*.

Finally, in order for this information to be useful to business process re-engineering, enterprise viewpoints (*e.g.* CIMOSA's function, organization, information, and resource viewpoints) must be established. The obvious approach is to simply adopt and encode in IML an existing enterprise modeling practice; there are many to choose from [19], [20], [21], [22]. A more interesting approach is to explore whether these (and other) enterprise modeling viewpoints may be synthesized through *multi-dimensional separation of concerns,* as described in [4]. The technique allows for the modularization and management of concerns that may be expressed in various model elements across various models.

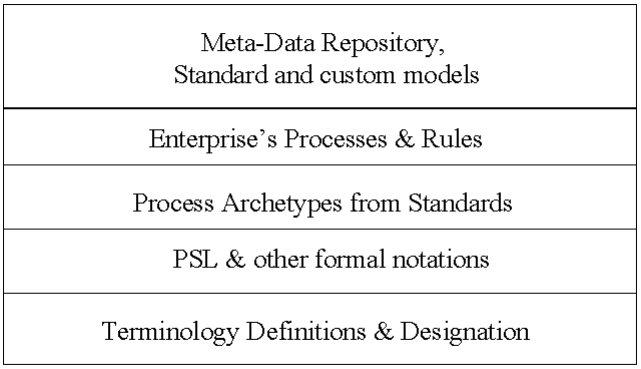

Figure 6 – Standard and custom environment information

V. Related work

In term of general perspective on the role that OMG technology can play in its solution, this work is inspired by D'Souza's paper [23].

The position on requirements engineering taken here is strongly influenced by the work of Zave and Jackson, [3], [24].

Ossher and Tarr [4] define a technique to modularize and relate project concerns. In addition to providing an additional dimension of coherence, the technique may provide a tractable means to manage reified refinements and concerns. The notion of concern in Ossher and Tarr's work differs somewhat from that here. Here concerns may be embedded within an enterprise model and have the full expressivity of first order predicate logic.

Nuseibeh [25] has investigated the reconciliation of viewpoints. The notions of refinement and concern used in this paper have similarity with their conception of viewpoints.

The approach described here does not define enterprise model viewpoints, as does CIMOSA [19], ARIS [20] and the Zachman Framework [21]. In this respect the approach more closely resembles work where enterprise viewpoints are defined ad hoc, to meet the specific requirements of the integration work at hand. Nissen *et al.* [22] have used ConceptBase [31] in that respect. None of the aforementioned work uses a meta-model of modeling technology, however.

VI. Conclusion

The paper outlines how a model repository built on a family-of-languages meta-model could be used to establish coherence among the project-specific models it manages. This coherence aids in the resolution of difficult problems that large-scale integration projects inevitably face. The key to the solution described is the embedding of an inter-model language in the repository. The language may be used to define a conceptual model that provides context for project-specific models and reify relationships among elements of

those models. When attention is paid to the separation of the IML-based environment model from project-specific models in the repository, and links are established between the two, the coherence resembles aspects of an enterprise model.

To date, we have just begun work towards validating this approach and refining our knowledge of the problems, having begun work on a meta-data repository and experiments with Powerloom [12], software implementing a description logic. It is likely that our final solution will differ substantially from what is described here. There is much work to be done. Specifically, the taxonomy of refinement and concerns must be studied further and elaborated upon. Secondly, the notion of enterprise model viewpoints in an IML-based environment model must be explored. Its tractability is crucial to business process re-engineering under the architecture. Additional work is required to determine whether the 'separation of concerns' techniques of Ossher and Tarr [4] might apply here.

Finally there are issues with respect to the management of refinements in the modeling environment. A broadly defined conceptual notion asserted in the inter-model language would have links to model elements in various models. Managing this complexity without taxing the user unreasonably may be challenging. The separation of conceptual and requirements modeling into IML code is a first line of defense against problems here. Multi-dimensional separation of concerns has relevance here also.

We plan to explore the approach in the domain of manufacturing systems integration, based on prior work in product data management [35], enterprise resource planning [33] or machine control [34].

REFERENCES

[1] S. Cook, "The UML family of languages," *UML 2000 - The Unified Modeling Language: Advancing the Standard*, Third International Conference, York, UK, October, 2000, Springer Lecture Notes in Computer Science, vol. 1939, 2000.

[2] A. Evans, S. Kent, and B. Selic, Preface of *UML 2000 - The Unified Modeling Language: Advancing the Standard*, Third International Conference, York, UK, October, 2000, Springer Lecture Notes in Computer Science, vol. 1939, 2000.

[3] P. Zave and M. Jackson, "Four dark corners of requirements engineering," *ACM Trans. Soft Eng and Meth*., Vol 6, No. 1, January 1997.

[4] H. Ossher and P. Tarr. "Multi-Dimensional Separation of Concerns using Hyperspaces," *IBM Research Report 21452*, April, 1999.

[5] International Organization for Standards, *ISO/IEC 10027 Information technology – Information resource dictionary system (IRDS) framework*, 1990.

[6] Object Management Group, *Unified Modeling Language Specification, Version 1.3*: ftp://ftp.omg.org/pub/docs/ formal/00-03-01.pdf, 2000.

[7] P. Denno, "Modeling Requirements for Self-integrating Manufacturing Systems," In *Proc. The 4th International Conference on Design of Information Infrastructure Systems for Manufacturing*, Melbourne, Australia, 2000.

[8] Federal Information Processing Standards, *Integration definition for function modeling (IDEF0)*, National Institute of Standards and Technology, Gaithersburg, Maryland, 1993.

[9] International Organization for Standardization, *ISO 18629-11: Process Specification Language Core.* ISO TC184/SC4 Working Draft, 2001.

[10] T. Halpin and J. Gray, *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*, Morgan Kaufmann Publishers, 2001.

[11] J.C.S. Do Prado Leite, and A.P.A Oliveria, "A client oriented requirements baseline," In Proc. *Second IEEE Intrnl. Symposium on Requirement Engineering*, IEEE Computer Society, ISBN 0-8186-7017-7, 108-115.

[12] Powerloom, http://www.isi.edu/isd/LOOM/PowerLoom/index.html.

[13] R.J. Brachman and J. Schmolze, An overview of the KL-ONE knowledge representation system," *Cognitive Sci* 9(2), 1985.

[14] T. C. Lethbridge, "Practical techniques for organizing and measuring knowledge," PhD. dissertation. University of Ottawa, November, 1994.

[15] J. Warmer and A. Kleppe, *The Object Constraint Language: Precise Modeling with UML*, Addison-Wesley, Reading, Massachusetts, 1999.

[16] T. Halpin, "Augmenting UML with Fact-orientation," In *Hawaii International Conference on System Science (HICCS-34),* Maui, January, 2001.

[17] J. K. Sharp, "Precise Meaning of Object Oriented Models," In *The Journal of Conceptual Modeling*, Issue Number 8, April, 1999. http://www.inconcept.com/JCM/April1999/sharp.html,

[18] IEEE P1600.1 Standard Upper Ontology (SUO) Working Group, http://suo.ieee.org/, May 2, 2001.

[19] Esprit Consortium AMICE (editors*) CIMOSA: Open System Architecture for CIM*, 2nd revised and extended edition, Springer-Verlag, Berlin, 1993.

[20] A. W. Scheer, *Business process engineering : reference models for industrial enterprises*, study edition, Springer-Verlag, 1998.

[21] J.F. Sowa and J. A. Zachman, "Extending and Formalizing the Framework for Information Systems Architecture," *IBM Systems Journal*, vol. 31, no. 3, 1992.

[22] H. Nissen, M. Jeusfeld, M. Jarke, G. Zemanek, and H. Huber. "Managing multiple requirements perspectives with metamodels.," *IEEE Software*, 12(6):37--48, 1996. 12

[23] D. D'Souza, "MDA - An Architecture for Modeling, Enabling Model-Driven Integration," http://www.catalysis.org/omg/index.htm, 2001.

[24] M. Jackson and P. Zave, "Deriving specifications from requirements: An example," In *Proceedings of the 17th International Conference on Software Engineering*. ACM, New York, 1995

[25] B. Nuseibeh, J. Kramer and A. Finkelstein, "A framework for expressing relationships between multiple views in requirements engineering," *IEEE Transactions on Software Engineering*, v. 20, n. 10, Oct., 1994.

[26] Object Management Group, *Model Driven Architecture A Technical Perspective*, http://cgi.omg.org/doc?ab/ 2001-02-05, February 21, 2001.

[27] D. Flater, "Impact of Model-Driven Architecture," http://www.omg.org/mda/

[28] M. Fowler and K. Scott, *UML Distilled.* Addison-Wesley, 2000.

[29] Object Management Group, *UML 2.0 OCL Request for Proposals*, http://cgi.omg.org/doc?ad/2000-09-03, September 18, 2000.

[30] Object Management Group, *Meta Object Facility (MOF) Specification, Version 1.3*: ftp://ftp.omg.org/pub/docs/ formal/00-04-03.pdf, March, 2000.

[31] M. Jarke, S. Eherer, R. Gallersdorfer, M. Jeusfeld, and M. Staudt, "ConceptBase – a deductive object base manager,"

[32] S. M. McMenamin and J. F. Palmer, *Essential Systems Analysis*, Yourdon Press, 1984.

[33] E. Barkmeyer and M. E. Algeo, "Enterprise resource planning systems in manufacturing,," in *Handbook for industrial engineering*, August, 2000.

[34] D. Flater, E. Barkmeyer, and E. Wallace, "Towards unambiguous specifications: five alternative job control models for an object-oriented, hierarchical shop control system," in *Proceedings of the 1999 ASME Design Engineering Technical Conference*, September 12-15, 1999.

[35] Object Management Group, *PDM Enablers*, http://cgi.omg.org/cgi-bin/doc?dtc/2000-06-02