

Perspective Models: A Mechanism for Achieving Interoperability Among Expressive, Personalized Domain Views

Kym J. Pohl

[CDM Technologies Inc.](#)

2975 McMillan Ave.

San Luis Obispo, CA. 93401

805-541-3750 x233

kpohl@cdmtech.com

<http://www.cdmtech.com>

Abstract—Accurate and expressive representation of the subject matter over which a context-oriented, decision-support system operates is fundamental to the effectiveness and longevity of the resulting solution. Often taking the form of an ontology, such extensive representational models, by their very nature, are rich in relationships and both coarse and fine-grained objects. It is, however, these qualities enabling rich expression that can significantly increase both the complexity of developing against these models as well as the potential for incurring undesirable performance issues. Further, due to the typically detail-oriented usage inherent in the software-based users (i.e., reasoning agents, etc.) of these models, it is important to recognize that a singular view of the world so to speak is not necessarily appropriate across the entire Ontology user base. In fact, in such highly expressive environments, it is critical to not only recognizing these distinctions in user perspective, but to, in fact, promote and exploit them. It is by acknowledging and consequentially supporting this perspective-based individuality among Ontology users that true representational accuracy and utility is achieved.

Traditionally, software-based users comprising decision-support systems have operated over a singular, common representation. However, in the Perspective Model-enriched environment presented in this paper¹, Ontology users are empowered with the ability to effectively perceive the world in accordance with individualized, native views. These views are then seamlessly inter-linked with one another to form a multi-Perspective Model of the target domain capable of supporting rich interoperability. Exclusively operating over personalized Perspective Models, users are not only shielded from the broad-scoped complexities inherent in the more omniscient concerns of the Ontology's entire scope but are also able to both view and interact with it in terms of more native representation.

To be effective, the concept of Perspective Models must be partnered with a supportive model development process. In addition to an explanation of the concept of Perspective Models, this paper also presents a purpose-built

development process that supports effective creation of the potentially numerous sets of models inherent in this type of expressive paradigm. The process offered in this paper effectively parcels the development of individual Perspective Models with the individuals possessing the necessary domain and use-case expertise. In this manner, the development process strives to significantly increase the involvement of the entire set of team members in the modeling activity, both capitalizing on user domain expertise in addition to increasing critical user understanding as well as acceptance of the representation over which their components will operate.

TABLE OF CONTENTS

1. REPRESENTING PERSPECTIVE.....	1
2. AN EFFECTIVE DEVELOPMENT PROCESS.....	5
3. CONCLUSION.....	6
REFERENCES.....	6

1. REPRESENTING PERSPECTIVE

Fundamental to context-oriented reasoning is the highly expressive representation over which intricate analysis is performed [8] [12] [13]. Often in the form of an Ontology, such elaborate descriptions form the foundation underpinning the effectiveness of context-oriented, decision-support systems. An Ontology in the scope of this paper^{1,2} is defined as a highly expressive, typically relationship-rich model of the potentially extensive subject matter over which software components, hereunto referred to as *users*, reason and otherwise operate.

¹ Copyright CDM Technologies Inc., 2008

The Significance of Perspective

Perspective is applied each time we as human beings perceive something. Although certainly at times aligning fairly closely across multiple observers, such perspectives are inherently unique to the individual. Housed within these individualized perspectives is valuable information describing how a particular topic is most suitably represented from a certain point of view. In addition, such perspectives also convey how a particular subject relates to other subject matter seen as relevant by the particular individual. Even when a concept or *thing* has a common basis among observers, individual perception is typically still biased toward personalized experiences and overall knowledge. Although at times a significant complication for meaningful interaction, such perspective is extremely significant to accurate representation as it is rich in descriptive context. For example, consider the following illustration involving the laptop on which this paper was written. In the case of a software system assisting within the initial manufacturing process, the laptop might be most effectively described in terms of its product-oriented nature. In this sense, the most suitable representation of the laptop would revolve around characteristics relevant to assembly, packaging, and other such manufacturing-oriented concerns. Further, relationships to customer orders and delivery schedules would also be important to represent. In contrast, however, characteristics explicitly describing the laptop's utility in authoring publications or developing software are fairly peripheral, if not completely irrelevant to the target manufacturing domain. However, such perspective may be quite relevant to, for example, the interests of marketing or perhaps even customer-support. Of course both perspectives are quite valid with respect to their individual areas of operation. However, both views would inevitably encompass some of the same subject matter (i.e., laptops) yet describe them in distinctively different manners. The problem arises when users of distinctly different representations of the same subject matter attempt to interact. This situation can produce a significant dilemma. Simply stated, the valuable context that is expressed within individualized perspectives can also significantly limit the ability for users to interoperate in a meaningful fashion (i.e., in terms of rich context).

However, despite the complications brought on by attempting to capture and exploit distinctive perspective, support for such personalized expression can significantly increase the quality of analysis performed by intelligent software agents operating within a decision-support environment. Perspective-enriched models can successfully capture not only the sometimes subtle distinctions among Ontology users, but by doing so can promote a more expressive description of each user's perception of their world. Unfortunately, due to the complexity inherent in identifying and supporting such subtleties and nuances, representation approaching this level of expression has traditionally been buried as implied assumptions within

convoluted business logic or simply omitted entirely. However, when appropriately represented and housed within the context tier of a collaborative environment, such expressiveness can not only be effectively exploited, but is also much more readily accessible to users.

Perspective Models

However, even with perspective sufficiently represented within the context tier, the ability for users of such perspective to interact in terms of their individualized views poses a substantially complex interoperability problem. The solution to this interoperability dilemma comprises three elements. The first focuses on the development of a singular, all-encompassing ontology referred to as a *Universal Model*. As the name implies, Universal Models are an attempt to develop an all-purpose, amalgamation satisfying all possible use-cases and perspectives. In this paradigm, each user would utilize the Universal Model as its primary *language* for interacting with other users. As a distinct strength of this approach, each user would essentially dialogue with one another in terms of a single representation promoting interoperability in a clear and concise manner void of any context-diminishing translation. Each user would essentially share the same *view of the world*. However, considering the complexity resulting from collapsing what could possibly be numerous perspective-oriented characteristics into a single description, the resulting model would be severely bloated and would most likely fail to adequately represent any one particular perspective, resulting in a model confusing to utilize.

The second, somewhat related attempt at solving this dilemma addresses the inevitable complexity of the Universal Model approach described above and offers a more delineated organization. In this approach, each particular subject matter is modeled in terms of its fundamental, intrinsic nature. The various perspectives applied to each particular subject are explicitly represented as individual model fragments. These perspective *sub models* are connected to the subject models they enhance using the *role analysis* pattern [3]. Such a connection can be conceptualized as something *playing* a variety of roles with each role representing a particular view on that subject. In this fashion, individual perspectives can be easily managed and clearly discernable from one another. In addition, this approach offers a degree, although limited, of encapsulation and isolation from irrelevant perspectives as users can isolate their interaction with a subject matter to those perspectives that are meaningful to them. Further, additional perspectives can be integrated in a manageable fashion through the incorporation of new roles-based model fragments. As a result, each subject is connected to model fragments describing the various contexts in which it can be viewed. For example, interaction with the aforementioned laptop subject from a manufacturing-oriented perspective may be in terms of a related *ManufacturedProductRole* model fragment. However, the problem with this approach

is that even though perspectives relating to the same subject matter are somewhat partitioned from one another, they remain integrated into a single model with no explicit management and depending heavily on diligent usage. As such, additional access control may need to be employed to truly isolate users to relevant perspectives. In addition, there is still the dilemma of whether or not a slight difference in two perspectives is worthy to warrant creation of an entirely new Perspective Model fragment. In practice, one would be tempted to collapse subtle differences in perspective into a single, overloaded model fragment, thus compromising accurate expression.

The third, more promising solution to supporting individualized yet interoperating perspectives introduces the notion of a *Perspective Model*. Based on a semi-stateful *façade* design pattern [5], Perspective Models allow context-rich subject matter to be viewed by inter-operating users in terms of individualized, native perspective. Perspective models may directly contain their content, derive it from some type of shared source (e.g., an Integration Model), or comprise a combination thereof. While state simply for local consumption is represented and maintained within the Perspective Model itself, derivation is used for material that is shared across users (i.e., the basis for collaboration). In the case of derived content, the function of the Perspective Model may, for example, be to apply more native terminology, structure, or other characteristics that more appropriately represent the manner in which the particular user wishes to see the world. In some cases such mappings, either uni-directional or bi-directional, may be fairly straightforward and easily describable through standard expression grammar. However, in other cases these mappings may be rather complex to the point of requiring customized behavior. In either case, such mappings can be effectively described in terms of a formalized language such as XSLT [1] [9] or CLIPS-based rule sets [6] [11].

Integration Model

As mentioned earlier, derivation is essentially the means for linking together multiple perspectives applied to the same subject matter. While there are a number of approaches to supporting such integration, it is critical that the individuality and bias exhibited by each Perspective Model is preserved in its native form. These models are essentially a user's most familiar and descriptive language with which to interact with the rest of the world (i.e., other users).

The approach presented in this paper to interconnecting disparate perspectives of the same subject matter employs the notion of an *Integration Model* in conjunction with the *façade* design pattern [5]. Although not a necessity, employing an Integration Model as a central *hub* from which interacting models are mapped in and out of avoids the *many-to-many* mapping paradigm inherent with a more

direct perspective-to-perspective connection. With this approach, a central, role-based representation of clearly delineated perspectives, not unlike the second alternative to integrating multiple perspectives described earlier, is developed as a well-structured and delineated combination of individualized perspectives related to the intrinsic subject matter they enhance. For example, the main subject of our earlier example might take the form of a *laptop* entity that can play the role of a *manufactured product*, as well as perhaps the role of a *software platform*. While the laptop entity would be focused on describing the subject's intrinsic nature, characteristics specific to each of these two perspectives would be housed within each related role.

As a further, diagrammatic description of this connection, Figure 1 describes a logistically-oriented Perspective Model linked to an Integration Model that presents a fairly neutral description of a conveyance. As an aside, note that conceptually such neutrality is not necessarily a prerequisite in that if the Integration Model were more heavily biased toward a particular perspective, it would simply imply that the Perspective Models might need to be more extensive and incorporate additional constraints. However, in the interest of clarity, this example employs a somewhat neutral Integration Model.

Central to the logistics perspective presented in Figure 1 is the notion of a *transport*. Although the logistics perspective may have knowledge of the entire set of conveyance types (i.e., vessels, vehicles, and aircraft) represented in the Integration Model, in respect to the logistics view, only vessels and rotary aircraft are considered candidate transports. In this situation, it would be valuable to represent this constraint in the Perspective Model employed by the logistics system while still basing such a biased view on the much more neutral representation of the conveyance offered by the Integration Model. As Figure 1 illustrates, representing such refinement can be accomplished by explicitly introducing a constrained notion of a *transport* in the logistically-oriented Perspective Model. According to the particular perspective, an abstract *Transport* is defined as taking two specific forms (*VesselTransport* and *HelicopterTransport*). At this point, it is immediately apparent that a *vehicle* is not a candidate to be a transport, from that perspective. In the context of this example, transports can only be *VesselTransports* or *HelicopterTransports*. The task now becomes linking this perspective together with the core Integration Model. Relating these two transport types to their conveyance derivation can be achieved in either an explicit or implicit manner. For illustration purposes, the definition of *VesselTransport* adopts the first method while *HelicopterTransport* employs the second. The first method defines an explicit, and exposed relationship between the *VesselTransport* and the core description of a vessel outlined in the conveyance section of the Integration Model.

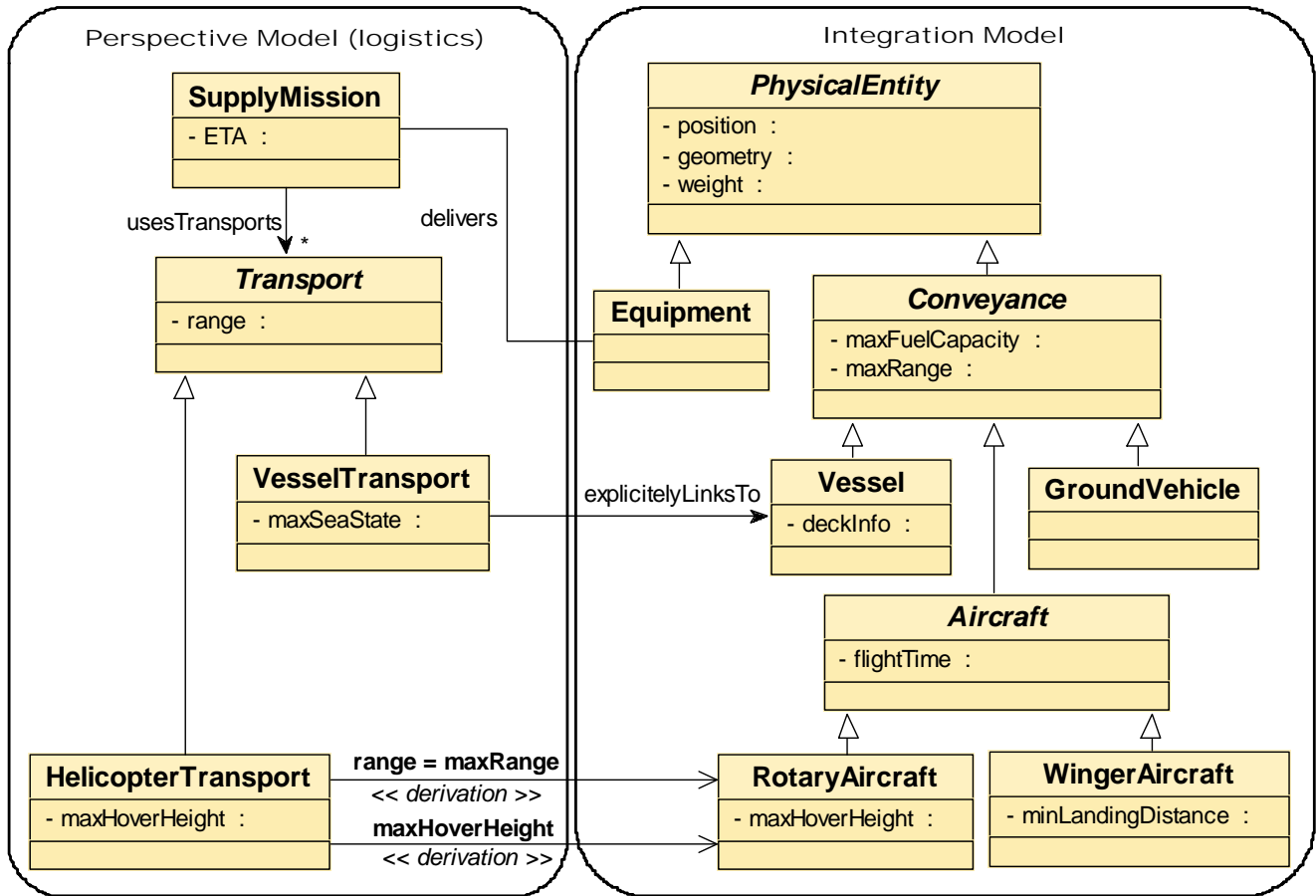


Figure 1 – UML [4] Diagram Illustrating A Logistics Perspective Model Deriving From A Relatively Unbiased Central Integration Model

Utilizing this approach, obtaining the core information relative to the corresponding *Vessel* from a *VesselTransport* requires both knowledge of their relationship in addition to a further level of indirection. For reasons of performance and representational precision, both of these requirements may not be desirable.

The second method, illustrated in Figure 1 using *HelicopterTransport*, overcomes both shortcomings inherent in the first approach. In this case, *HelicopterTransport* is represented in terms of a façade, or filter of sorts, which transparently connects this biased view to the core *RotaryAircraft* description housed within the Integration Model. That is, each attribute of *RotaryAircraft* relevant to the notion of a *HelicopterTransport* is explicitly declared within the façade. For example, since the maximum range of travel is relevant to the definition of a *HelicopterTransport* the *maxRange* attribute of *RotaryAircraft* (inherited from *Conveyance*) is subsequently exposed in the *HelicopterTransport* façade. By virtue of being declared as a derived property, any access to such an attribute would be transparently mapped to the corresponding attribute(s) housed within the Integration

Model. In the case of the *range* attribute of *HelicopterTransport*, access is transparently directed to the inherited *maxRange* attribute of *RotaryAircraft*. Notice also the use of alternative terminology over that used in the Integration Model (i.e., *range* vs. *maxRange*). It should also be noted that the derived nature of a façade attribute is not limited to mapping to a single attribute. Rather, the value of a façade attribute may also be derived through specific behavior, perhaps a calculation or algorithm based on the values of multiple attributes residing across several Integration Model objects. In either case, the fact that the value of the façade attribute is derived, and not originating locally, is completely transparent to users of the *HelicopterTransport* Perspective Model object.

Yet another perspective-oriented enhancement to the core Integration Model illustrated in Figure 1 is the notion of a *SupplyMission*. Being a fundamental notion of a logistics perspective, a supply mission essentially relates equipment in the form of supply items to the transports by which they will be delivered. Once again, the definition of a logistics-specific notion (i.e., supply item) is derived from a notion defined in the Integration Model (i.e., equipment). In this

case, an explicit relationship is declared linking *SupplyMission* to zero or more *Equipment* items. From the perspective of the logistics system equipment scheduled for

delivery is perceived as items to be supplied, the term *supplyItems* is a more appropriate nomenclature. Such

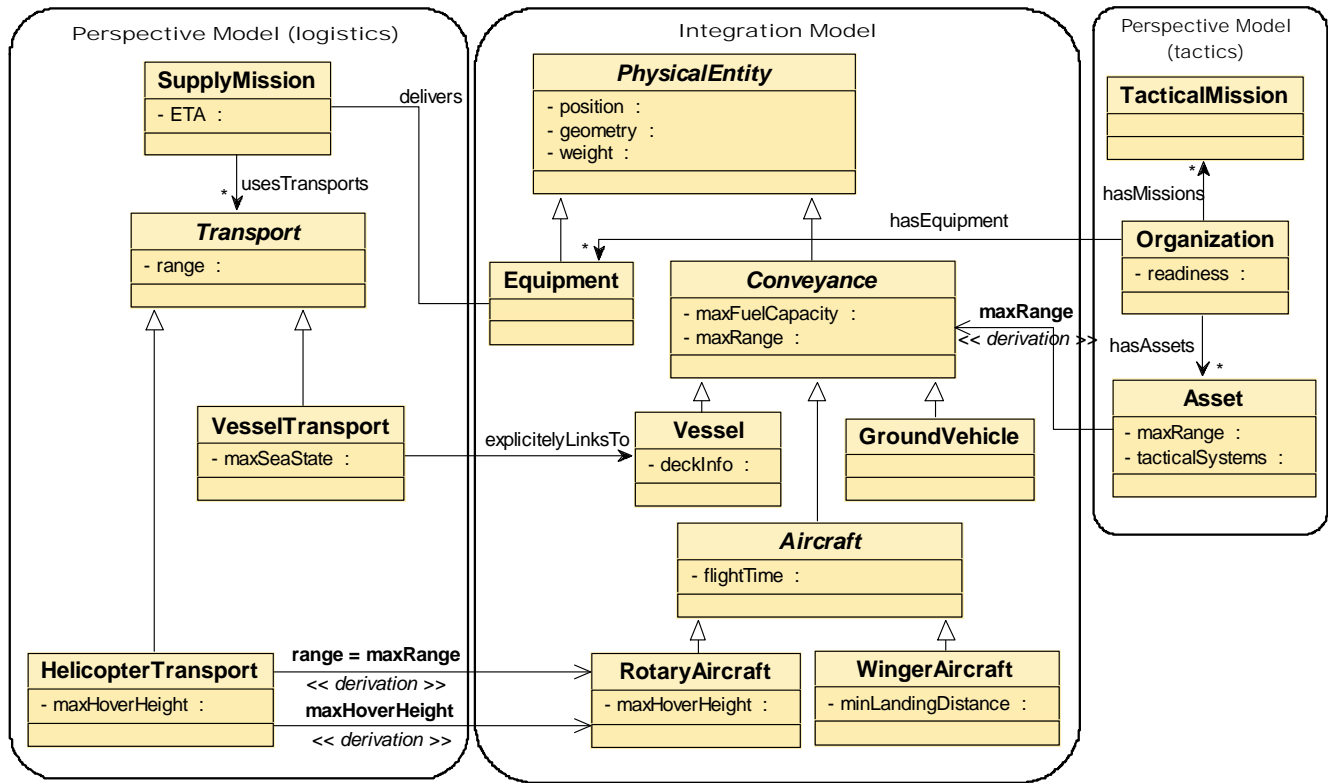


Figure 2 – UML [4] Diagram Illustrating Two Disparate Perspectives Connected Via A Central Integration Model

enhancement to the innate descriptions provided by the Integration Model demonstrates the ability of a Perspective Model to essentially overlay new notions (i.e., supply missions) over existing intrinsically-described subject matter (i.e., equipment and conveyances). To further illustrate how multiple, potentially diverse perspectives can be effectively integrated to support meaningful interoperability, Figure 2 elaborates on the example by introducing an additional perspective on the core subject matter. The additional perspective is concerned with a more *tactical* view of the domain. Collaboration between these two perspectives is enabled by the common Integration Model from which many of their notions derive. A conveyance is still a conveyance whether viewed in the context of logistics operations or tactical command and control. Although both users may discuss a conveyance from partially disparate perspectives, both can effectively collaborate about a particular conveyance in terms of their own native, biased perspectives.

2. AN EFFECTIVE DEVELOPMENT PROCESS

Perspective models can be a powerful means of capturing and exploiting the expressive nature inherent in individuality. However, to arrive at an effective approach,

such a method must be accompanied by a complimentary development process. Traditional approaches to domain model development have typically involved a dedicated knowledge engineer, or group of such individuals, whose task it is to produce a well structured representation of the target domain(s). Following creation of such a model, component developers design and implement functionality in terms of, or at least in a form that is compatible with, this representation. The problem inherent in this approach is essentially twofold. First, while model development is usually driven by a focused study of the domain this study typically does not include the specific use cases of its intended users. After all, the primary purpose of the representation sustaining a context-oriented, decision-support environment is to effectively support the data, information, and knowledge needs of its users. To ensure effective support of these activities, such implicit use-cases should be one—if not *the* most significant—force that drives model development.

The second pitfall of a conventional modeling approach also deals with the potential disconnect between a subject matter representation and its users. However, in this case the problem manifests itself at a more humanistic level. Critical to the successful application of an often fairly complex

representation is the degree to which project team developers embrace, and are able to become familiar with, the various structure and semantics comprising the model. This is especially true in the case of reasoning-based, decision-support systems which tend to operate over complex, highly expressive contexts. To effectively exploit the expressive nature of context-enriched models requires developers to both understand such representation at a semantic level as well as embrace the manner in which it represents their subject matter interests. Many systems have fallen far short of their potential, sometimes to the point of complete failure, due to a lack of team member understanding and *buy-in* to the manner in which their domain(s) are represented.

The development process offered in this discussion addresses this disconnect by significantly increasing the involvement of model users with the actual model development activity itself. There are a number of benefits to such team member inclusion. First, as component developers research and design their solutions (i.e., software components), they essentially acquire a considerable amount of expertise and knowledge regarding relevant domain(s). Such familiarity goes beyond a fairly deep understanding of the semantics of relevant subject matter and includes valuable insight into the precise means by which particular functionality might most effectively view such content. It is the identification and subsequent capture of such individualized expression that produces a truly accurate representation. Since the focus is on capturing native perspective and bias, there is no need at this stage—in fact it would be potentially polluting—to be concerned with the degree to which these models align with each other. Narrowing the scope of individual Perspective Model development not only promotes the capture of true individuality, but is also a significantly less complex task than developing a singular, all-encompassing model supporting the entire set of interconnected perspectives (i.e., Universal Model). This less complex modeling environment has a direct impact on the amount of expertise and experience required for effectively developing these personalized Perspective Models. While good modeling practices are still quite important in this process, they can be applied within considerably less complex environments by individuals who may not have the modeling depth of an experienced knowledge engineer. Further, familiarity with model structure and subsequent semantics undoubtedly leads to a significantly stronger bond between component developers and the subject matter representation over which their components operate.

Development of the Integration Model itself is a notably more involved task than that of developing the various Perspective Models. Development of the Integration Model involves the analysis of each Perspective Model with an eye for both identifying and abstracting subject matter existing across the multitude of user perspectives. Further, this

subject matter must be modeled in a manner that maintains overall consistency and integrity as well as promotes expandability as additional inclusion of additional content is needed. Considering the complexities involved in this task, in addition to the demand for being both knowledgeable and comfortable with applying various intricate analysis patterns, this activity typically requires a highly experienced, expert modeler. As such, this activity might become the main area of focus for the expert knowledge engineer(s) who have traditionally been responsible for the entire modeling activity.

The final component to building the Integration Model is to describe the derivation logic that effectively ties the various Perspective Models with the central Integration Model. Coupled with some type of code-generation facility capable of managing implementation concerns, such derivation specifications can be designed, communicated, and maintained at the modeling level. Similar to development of the actual Integration Model itself, development of these mappings will likely also require the skills of an experienced knowledge engineer.

3. CONCLUSION

To obtain truly accurate, expressive representation, individual perspective must be specifically captured based on the use-cases of its immediate user(s). Interoperability within a diverse, perspective-enriched environment must support meaningful interaction between users that preserves this individualized perspective. Applying Perspective Models interconnected via a unifying Integration Model effectively supports these two objectives. Further, employing a development process where Perspective Model development directly involves the very users themselves leads to a more precise and expressive representation while significantly improving the representation's effectiveness through increased user familiarity and imperative model adoption.

REFERENCES

- [1] Cagle, K., M. Corning, J. Diamond, T. Duynstee, O. Gudmundsson, M. Mason, J. Pinnock, P. Spencer, J. Tang, A. Watt, J. Jirat, P. Tchistopolskii, and J. Tennison, "Professional XSL", Wrox Press Ltd., Birmingham, UK., 2001
- [2] Daconta M., L. Obrst and K. Smith, "The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management", Wiley, Indianapolis, IN., 2003
- [3] Fowler, M., "Analysis Patterns: Reusable Object Models", Addison-Wesley, Reading, Massachusetts, 1997.

- [4] Fowler, M., "UML Distilled: Applying the Standard Object Modeling Language", Addison-Wesley, Reading, Massachusetts, 1997.
- [5] Fowler M., D. Rice, M. Foemmel, E. Hieatt, R. Mee, and R. Stafford, "Patterns of Enterprise Application Architecture", Addison-Wesley, Reading, Massachusetts, 2003
- [6] Friedman-Hill, E., "JESS In Action", Manning Publications Co., Greenwich, CT, 2003
- [7] Garshol L. and G. Moore (eds.), "The XML Topic Maps (XTM) Syntax", JTC1/SC34:ISO 13250, July 22, 2002, (www.y12.doe.gov/sgml/sc34/document/0328.htm)
- [8] Giarratano J. and Riley G., "Expert Systems: Principles and Programming", 2nd Edition, PWS Publishing Company, Boston, MA.
- [9] Hunter D., C. Cagle, D. Gibbons, N. Ozu, J. Pinnock, and P. Spencer, "Beginning XML", Wrox Press Ltd., Birmingham, UK., 2000
- [10] Karsai G., "Design Tool Integration: An Exercise in Semantic Interoperability", Proceedings of the IEEE Engineering of Computer Based Systems, Edinburgh, UK, March, 2000
- [11] NASA, "CLIPS 6.0 Reference Manual", Software Technologies Branch, Lyndon B Space Center, Houston, Texas, 1992
- [12] Pohl J., "Information-Centric Decision-Support Systems: A Blueprint for Interoperability", Office of Naval Research (ONR) Workshop hosted by the CAD Research Center in Quantico, VA, June 5-7, 2001
- [13] Pohl J, A Chapman, K Pohl, J Primrose and A Wozniak, "Decision-Support Systems: Notions, Prototypes, and In-Use Applications", Technical Report, CADRU-11-97, CAD Research Center, Design Institute, College of Architecture and Environmental Design, Cal Poly, San Luis Obispo, CA, January, 1997