Architecture Driven Information Sharing

Extracted from the SOPES IEDM Specification Annex A: Modeling Profile Description

The following information was directly extracted from the Object Management Group (OMG) Shared Operational Picture Exchange Services (SOPES) Infromation Exchange Data Model (IEDM) specification. In the fall of 2010, The Object Management Group is expected to adopt the Shared Operational Picture Exchange Services Specification (SOPES) IEDM. The SOPES IEDM specification formalizes a platform independent set of data patterns for the construction, parsing and processing of JC3IEDM semantics for situational awareness and collaborative planning. The data patterns apply directly to a set of transactions for the MIP Joint Consultation, Command and Control Information Exchange Data Model (JC3IEDM: version 3.1 c ratified December 2007). The specification provides this set of data patterns as building blocks for the exchange of information that is applicable to a wide range of operational communities, including:

- First Responders (e.g., Police, Fire Department and Emergency Medical Personnel);
- Government Agencies (Federal, Provincial/State, and Municipal);
- Non-Governmental Organizations (NGOs);
- Other Government Department (OGD);
- Private Volunteer Organizations (PVOs);
- Para-military and security agencies; and
- Military (Joint, land, maritime, air, space and coalition).

These communities have comparable requirements for shared situational awareness, and collaborative planning. Their operations are increasingly crossing organizational, agency and national boundaries. The participating organizations are required to collaborate on asymmetric real-time operations such as: Crisis Response, Disaster Recovery, Humanitarian Aid, Sustainment and Support Operations, Public Health and Safety, Stability Operations and Homeland Security. The scope, complexity and frequency of these operations are presenting significant communication challenges. The SOPES specification provides a core set of information patterns that have the potential to bridge evolving community semantics and ontologies.

However, the SOPES modelling profile can be used to address information requirements beyond the JC3IEDM. The profile has been recently integrated into the second version of the Unified Profile for DODAF and MODAF to extend the abilities to architecture frameworks to specify information sharing and information protection policies within an enterprise, system of systems or systems environment. Annex A to the SOPES specification describes this modelling paradigm.

A.1 Overview

The modeling approach used in this specification describes a set of **reusable information patterns** (building blocks) for a structured information store; in this case the JC3IEDM. The approach is intended to specify the operational policies for the composition, construction, processing and protection of information composites (or aggregates or business objects) as they are shared within and between operational nodes (e.g., systems, applications or services).

The approach encompasses the following architectural elements:

Contract – A contract represents a grouping of semantics and information flow controls which specify a formal information sharing agreement between two or more operational nodes or participants in a domain or community (e.g., Community of Interest [CoI]). Although provided and described in the approach, this element is not a normative component of this specification which seek to focus on the transactional information patterns for the JC3IEDM, the contracts and semantics are deemed the purview of the operational communities such as MIP.

Semantic – A semantic represents the build policy for an information composite or data composite that is specified as meaningful to participants (applications, systems and users) in a particular domain or community. Only exemplar semantics are provided in this submission as guidance to the design and development communities.

Transactional - A Transactional represent the build policy for a reusable information building blocks, often realized as business objects comprising the community logical data model, for which there is likely also an underlying information or data store; they maintain the referential and data integrity of that store. Transactionals form the core of this specification.

Wrapper - A Wrapper directly maps to a data instance (e.g., row of data in a database application) in the logical data model and the physical data model.

Entity - An Entity is a Class mapping directly to the Physical Model specification for the underlying datastore.

Figure A.1 illustrates the proposed relationship between four architectural views of the UML Profile for DODAF, MODAF, NAF and DNDAF (OV-2, OV-3, OV-7, and SV-1 1). These views combine to describe the flow and language of communication within the enterprise, operational environment of system depicted by an architectural model.

The OV-2 identifies the flows of Resources (material, energy, organization, services, and/or information) between operational nodes which fall into the context of the architectural model. The flow of these need resources are realized on a "needline" between two or more operational nodes. Information flows realize the exchange of information-composites, which represent the aggregation of information and data elements described in the OV-7s and SV- 11s.

The OV-3 characterizes the flow of the information composites by specifying frequency, timeliness, safeguards, quality, etc. for each information flow (Information Exchange Requirements [IERs]).

The modeling approach aligns information exchange requirement or information flows (OV3) through to the logical and physical information definitions (OV-7 and SV- 11 respectively). The models establish policies (or rules) describing the logical construction of composite information from the information and data elements defined in the OV-7s and SV-1 1s. Each subtended element is built into a construction plan to systematically provide the information specified on the needline. The models are intended to provide traceability between the IERs and the application logic used to combine information and data elements of the information stores.

The contracts group the semantics of the community into information sharing agreements. Providing a separation between the agreements and semantically complete information-composites makes the semantics architecturally reusable components.

Each contract (information sharing agreement or ontological commitment) comprises one or more semantics (i.e., a COI exchange pattern with a defined

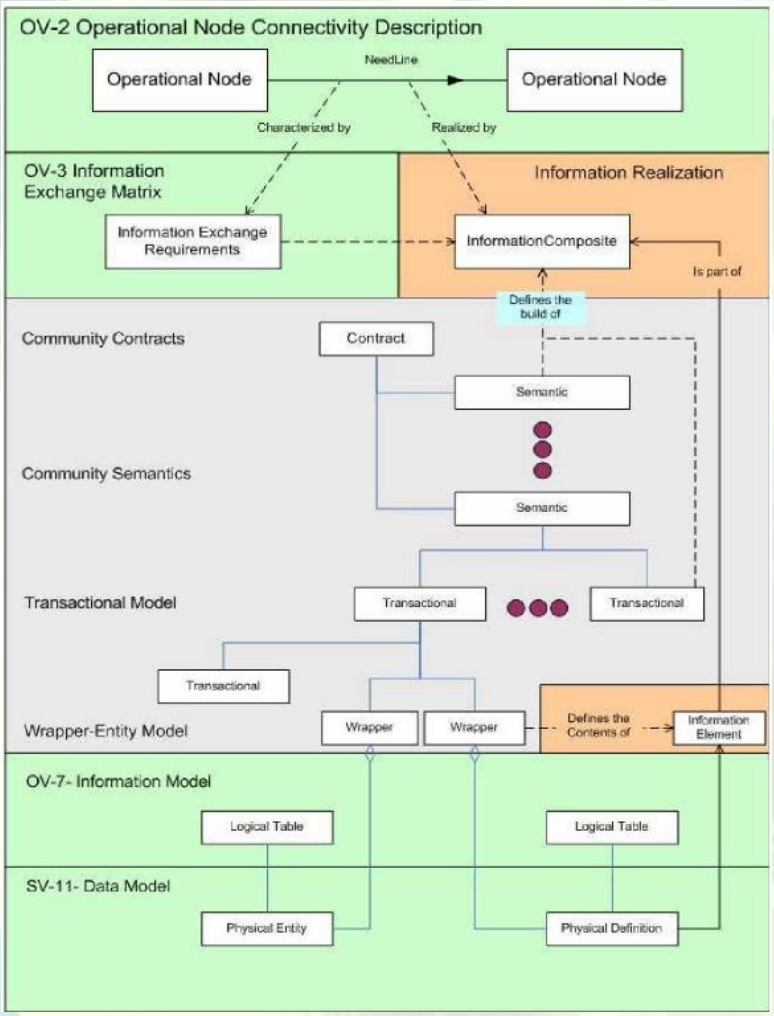

Figure A.1 - Alignment to the UPDM

meaning and purpose), which are specified by the participants to define information of relevance to their community. Each semantic is composed of one or more "Transactionals," which specify the logical information elements to be exchanged and how they are combined to meet the semantic requirements of the community.

The "Wrappers" represent the bridge between logical element of the transactional patterns and the physical data definitions SV-1 1, the Data Model. At the semantic, transactional, and wrapper levels there may also be formal domain rules and constraints that must be honoured by the parties to the contract.

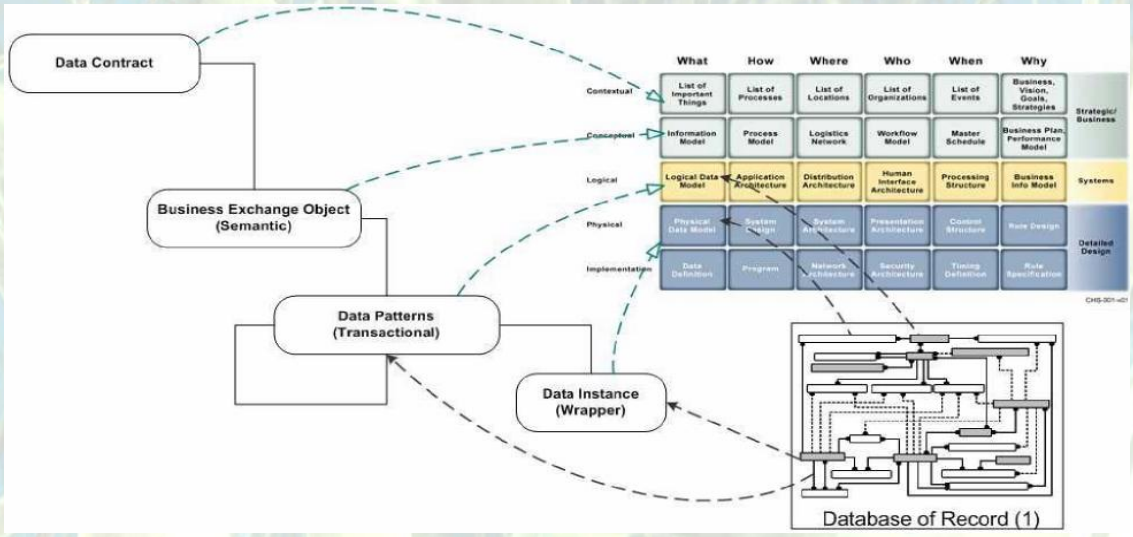

Figure A.2 - Alignment to the Zachman Framework

A.1 .1 Other Architecture Frameworks

A separate alignment can be presented for other architectural frameworks. However, with the OMG's current focus on the UML Profile for DODAF and MODAF it seemed reasonable to present the architectural alignment to the related frameworks (e.g., DODAF, MODAF, NAF, DNDAF, PSAF and others). As an example the alignment to the Zachman Framework is depicted in Figure 2 (above).

A.1.2 Model Extensibility

Later in the document the modeling approach will be extended to model domain filters and attribution which extend the policy models for information tagged with security, Quality of service and other information considerations.

A.1 .3 Modeling Objectives

The following objectives are critical to developing the concepts for policy based information interoperability:

- A modeling profile based on UML;
  - Explicitly capture, as part of architecture, the business rules for the export, transform and load processes, which are typically embedded in middleware applications. These include:
  - Community semantics,
  - Data store transactions,
  - Transformations (re-usable data patterns),
  - Data suppression filters, and
- Domain business rules.
- Assure that the concepts captured in the model enabled Model Driven Architecture (MDA) transformations to executable policies, which were alterable in the operational environment;
- Make the models useful and meaningful to stakeholders and users;
- Alignment with evolving architecture frameworks;
- Provide full traceability to requirements; and

- Design for change.

In an object environment (e.g., OO DB or object layer), support objects can be used (with a single existence) by multiple information-composites (semantics and transactions) providing a highly efficient use of information. Traditional approaches use a single information instance per composite causing increase memory and processing (e.g., data synchronization). Using the multi-use approach enables "event-driven global update." A single data change (new instance of data/information) can initiate the build and release of all transactionals and semantics in which the element is contained.

Within the context of data, information and knowledge management, ontology is defined as an information model describing a set of concepts within a domain of interest and the relationships between those concepts. This specification describes a set of information exchange concepts for ECM situational awareness and collaborative planning. The IEDM describes a set of information and knowledge patterns based on JC3IEDM-compliant transactions and information elements (i.e., data entities).

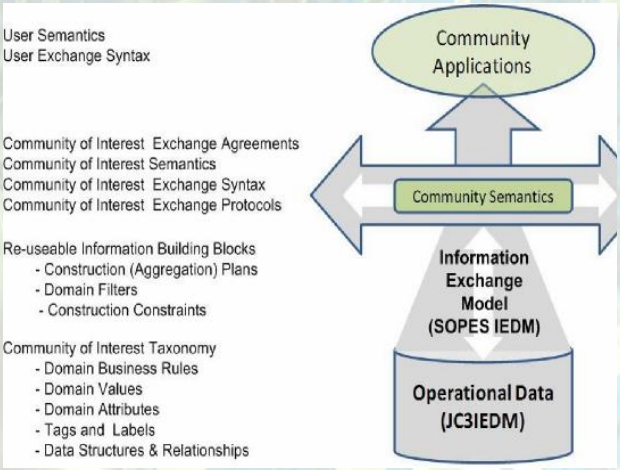

Figure A.3 - SOPES IEDM Scope

The Information patterns (Chapter 10 and 11) describe:

- Individual information elements;
- Classes: sets, collections, or types of objects;
- Attributes: properties, features, characteristics, or parameters;
- Relations: ways that objects can be related to one another, for data storage and in the construction of semantics (meaningful data object: this specification); and
- Events (watch points): changes to the data environment (e.g., attributes or relations) that trigger an exchange of information.

The specification describes a set of policies for constructing and interpreting information exchanges using reusable architectural components (information building blocks) aligned directly to commonly used architecture frameworks aligned directly to commonly used architecture frameworks as illustrated in Figure A.1 and Figure A.2.

A.1.4 Modeling Concept

The class models describe the policies (or rules) for processing information datasets; and aligning the datasets to the underlying data schemas. The objective of the models is three fold: 1) explicitly capture these key business rules as part the enterprise and System architectures; 2) retain corporate knowledge and understanding; and 3) separate the business rules from the underlying middleware application. Meeting these objectives, this modeling approach delivers auditable systems with increased interoperability, portability, and assurance.

As illustrated in Figure A.3, the semantics, transactionals and wrappers document a set of policies for the processing of reusable informational building blocks that align the information Exchange Requirements specified in an information exchange requirements to the information schemas underlying the operational environment.

A "Semantic" represents a set of policies for the construction or marshalling of information objects (i.e., a dataset) that is meaningful to the community (e.g., applications, systems, and users that form the context of architecture Model). The semantic is the uppermost concept in the ontological structure. When enforced by a system or application, a semantic realizes a complete information object (e.g., message payload) that provides a clear and consistent meaning for the community.

A "Transactional" specifies the policy (or rules) for the construction or marshalling of reusable information sets (e.g., realized as information objects) derived from the underlying logical model and associated physical data store(s). These plans form a set of informational building blocks that encapsulate semantics of the stores and set the rules for semantic completeness. The Transactionals also assure a semantically consistent treatment of information as it transitions in and out of the data store.

The term “Transactional” was adopted to align its core concepts with that of a database transaction - a concept well understood by the data and information management communities. The base transactionals would encompass the referential and data integrity of the datastore(s). The transactions are combined to complete the semantic requirements of the community. When enforced by information systems and applications the transactionals realize composite information sets needed to complete one or more semantics.

The “Wrappers” form the foundation of the modeling approach. The wrappers are a logical representation of instances of information elements that can be held within a data store. Each wrapper represents a single instance (or row of data) of data from the underlying store.

A.1 .5 Realization of Information

The models describe the policy (steps in a process) for systematically constructing or processing fused information sets (semantics). By definition the semantics ensure that the content exchange conforms to agreed community information patterns or semantics. It is important to understand that the models represent the specification for the aggregation or marshalling information - it is not the information carried on the needline; the actual information carried is referred to as an information-composite.

Definitions:

- Construction or Build: The process of aggregating information and data elements into their composite structures.
- Marshalling: The process of de-aggregating information composites and storing the information and data elements in their specified information or data stores.

A.1.6 Pattern Reuse

The modeling promotes the reuse of subtended elements and composites:

- An InformationComposite at the Semantic level can be reused to fulfill multiple commitments (Contracts), which use different messaging standards (e.g., National ADatP-3, OTH-Gold, MIP PDU).
- An InformationComposite at the transactional level can be reused within multiple transactional and semantics.
- An informationElement can be reused in multiple InformationComposites such that a single change (i.e., new data) cascades through each of the informationComposites enclosing the element; resulting in the updating of every contract and semantic holding the data.

The model approach supports derive information patterns (Figure 4) that enable concepts like event driven global update of all InformationComposite enclosing single instances of data enclose in well specified semantics.

A.1.7 Modeling Elements

Figure A.5 illustrates the basic modeling elements used in the models and the meanings applied to them. It is evident from the limited and standard set of modeling elements that the core concepts are not overly complicated and supportive of a broad community of practitioners.

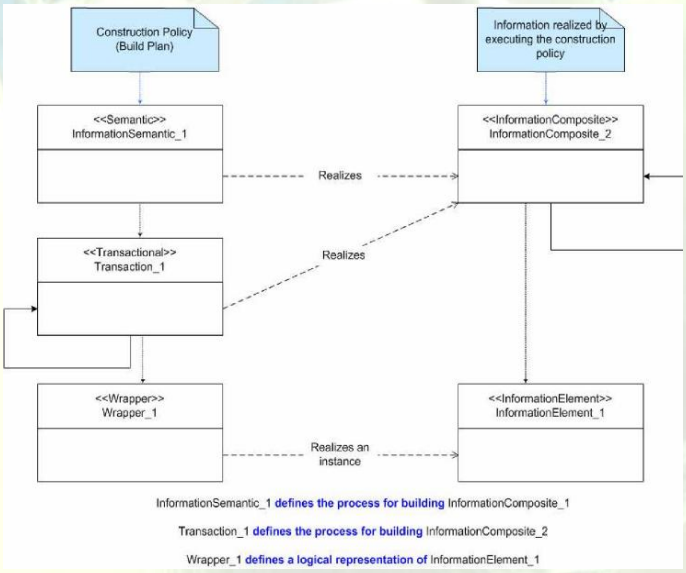

Figure A.4 - Realization of Information

The modeling approach uses UML Class Diagrams to identify all the subtended Classes of a Semantic or Transactional. Those stereotyped as “Transactional” are decomposed on a secondary class diagram. This modeling style aids in the readability of the models and simplifies each model element. Typically, the “Diagram Name” matches the “Enclosing Class” name whether a Contract, Semantic or a Transactional. Again this is for readability and publishing of the model.

A.1.7.2 Classes

Core modeling concepts: contracts, semantic, transactions and wrappers are included in the class diagrams as class stereotypes (Figure A.6).

Navigating the arcs of the class diagrams defines the construction plans for each information aggregation (i.e., transactions and semantics).

Classes fall into two categories on each diagram:

- Enclosing Class
- Subtended (Support) Class

Each class diagram identifies policies (rules) for building reusable information composites in the runtime environment. **A.1.7.3 Enclosing Class**

A.1.7.3.1 Overview

The “Enclosing Class” is the focus of a diagram and encapsulates the policies associated with the aggregation of information at runtime. Each Enclosing Classes realizes an object that encloses the aggregate of information from each of its subtended classes.

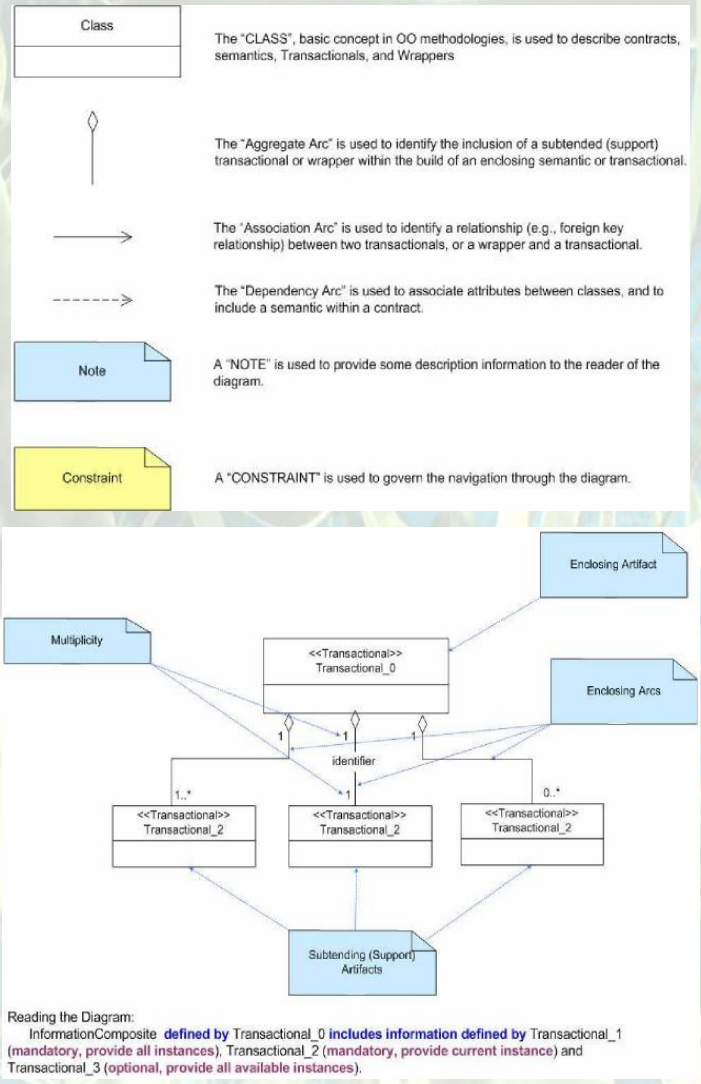

Figure A.6 - Modeling Elements

On the diagram, the enclosing class is the one to which the white diamond symbol on the association line is attached. The modeling style has one enclosing class on each diagram, which typically shares the same name and the diagram title.

In a runtime environment, semantics and transactions are only instantiated in response to a data event, and only persist for the period needed to construct or marshal the information-composites specified. The information aggregates, enclosing the information element / data event, are built in response to that data event.

Semantic and transactionals do persist their reference and policy data patterns comprising the community semantics. This informs the environment of the information instances active in the particular operational domain. These elements are persisted until explicitly removed from the systems’ or applications’ domain. This concept of persistence applies to both the semantic and transactional objects.

The information or leaf-node elements of the information patterns are persisted in the operational domain.

A.1.7.3.2 Identifying Class

In each diagram, there exists one and only one subtended class that is labeled as the “Identifier.” The Identifier indicates that the class on the labeled aggregation holds data that identifies which instance data is included in the build or aggregation. This information typically includes Database Keys or Unique Identifiers of some venue.

A.1.7.3.3 Subtended Class

The "Subtended (Support) Class" represents those classes, which are included within the build plan of the "Enclosing Class." Each subtended class is linked to the enclosing class through an enclosing aggregation arc. Subtended classes can be Transactionals or Wrappers.

A.1.7.4 Containment (Aggregation) Arc

The aggregation arc is read with inverse logic. In the information (/data) environment - the enclosing Class only exists if the mandatory subtended Classes exist. UML traditionalists would read the arc in the opposite direction. However, the models describe a build or construction policy for aggregated information sets, which require the existence of their subtended (support) classes to meet their semantic rules.

If mandatory subtended objects (identified through its multiplicity) do not exist, then the enclosing object cannot form or build and the policy fails. If optional subtended objects (identified through its multiplicity) do not exist, then the enclosing object builds with the information held by the existing subtended object.

As illustrated in Figure A.7, a single subtended element can be contained by multiple enclosing classes. This specifies that a change in that subtended object cause both enclosing objects to build at runtime. By cascading this concept, the models establish policy for event driven global update capability - one data event cause all semantics enclosing that subtended object to build and, if meeting there semantic requirements, and be released and fulfill ontological commitments of the participating communities.

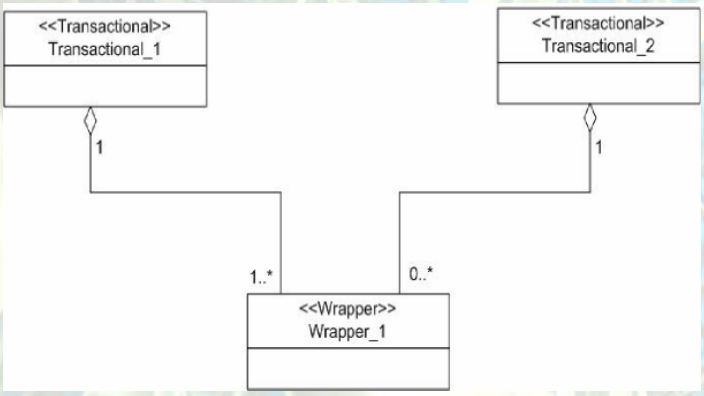

Figure A.7 - Single Instance Data

In addition. As illustrated in Figure A7-1, the containment arcs can contain a qualifier on the association which acts as a fixed fileter during the construction (aggregation) of a dataset under the prescribed pattern. These filters restrict the the collection of data to those datasets whose attribute ('attributeName') has a value of 'properValue'. E.g., self.securityLevel = "unclassified". Filters (qualifiers) are used selectively include or exclude information instances based on specific domain value instances at runtime.

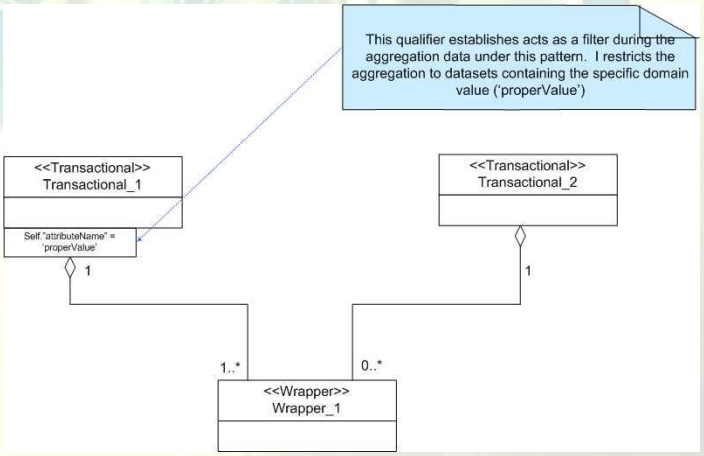

Figure A.7-1 – Addition of Filters

The formal SOPES Model provided in Section 10 does not caontain qualifiers as they are used to refine the model to specific operational requirements. With respect to the formal SOPES Specification filters were identified as extensible features.

A.1.7.5 Dependency Arc

The dependency arc is used in the contract specification to identify the relationship between the contract and the semantics, where a change in the semantic affects the semantics of the contract - resulting in the exchange of information. The arrow representing a dependency specifies the direction of a relationship, not the direction of a process.

A.1.7.5.1 Association

Navigable associations indicate that there is a relationship present between the associated entities in the underlying data store. Where an association is made between a Wrapper class and a Transactional class it is understood that the relationship exists between the Wrapper and the Identifier of the Transactional class.

A.1.7.5.2 Identifier

There exists on and only one "identifier" on each semantic or transactional diagram. The "identifier" identifies the subtended class that holds data elements needed for the construction of semantically complete information composite. This class would contain, as a minimum, the base Global Unique Identifier (e.g., Database Key, foreign keys or unique identifier) that would differentiate which transactional or wrappers (information element instances) are included in the construction of the composite (e.g., foreign key relationships).

A.1.7.5.3 Multiplicity

Multiplicity is presented on the aggregations to identify:

- The optionality of the subtended class;
- The number of information instances to be included in the construction of the information composite specified by the composite class (e.g., transactional or semantic). The multiplicity of the composite class is always "1."

Table A.2 - Multiplicity

Multiplicity Indicators		
Indicator	**Meaning**	
0. .1	Zero or one	Optional
1	One only	Mandatory
0..*	Zero or more	Optional
1..*	One or more	Mandatory
0. .n	Zero to n (where n > 1)	Optional
1. .n	One to n (where n > 1)	Mandatory

A.1.7.6 Constraint

The Constraints, Figure A.8, govern the construction for the composite information object. There are three areas where the modeling includes explicit constraints:

- Navigation constraint is used to constrain the inclusion of branches of the semantic tree based in specific domain value instances at runtime. Navigation constraints are primarily used when dealing with generalizations in the underlying data model (e.g., to select a specific subtype). The use of variable based constraints that apply only at runtime enables the selection of the specialization at runtime - allowing for variations in the semantic based on context. The OCL used in the models guide the inclusion of aggregations in the construction sequences of the defined patters and not intented to ne executable.

- Domain Rules are used to govern the allowable combinations of domain values in the underlying datastore (not illustrated). Domain rules can be contained within a single wrapper (entity / table) or cross tables. Domain rules are captures within the annotations of the classes.

Constraints are modeled in Object Constraint Language (OCL). In the future constraint definitions may be modeled using the structured English or Semantic Business Vocabulary and Rules (SBVR). To properly interpret a constrained aggregation, it is intended that the constraint be evaluated before its multiplicities. Should the constrain fail, the multiplicity is implicitly evaluated a zero.

For all instances of the constrained navigation the initial element 'self' is the enclosing Transactional, and in the case of the example from diagram A.8 self refers to 'InformationTransactional_1'. The second element of the constraint is the Wrapper instance which must be a directly subtended element of the enclosing Transactional, and in this case the Wrapper instance is 'Wrapper_1'. The third element is the named Wrapper Attribute featured on the specified Wrapper, and in this case this element is 'catCode'. The final element is evaluated against the constraint, and in this case the value is 'domainValue'.

A.7.7 Tagged Value

A tagged value is a combination of a tag and a value that gives supplementary information that is attached to a model element. A tagged value can be used to add properties to any model elements and can be applied to a model element or a stereotype. These tags are used to identify to the implementer specific information about the design pattern not carried with the attributes of the runtime object.

The tagged values used in the model include:

- isIdentifier identifies the start point for the build of an information composite or artifact. There exists one "identifier" per enclosing class.
- isWatchPoint both identifies the start point for the construction of the semantic contained in the model and identifies to the runtime what triggers the build of a semantic (data change in the subtended object); this tagged value, is a Boolean and assigned to an aggregation arc within a class model.

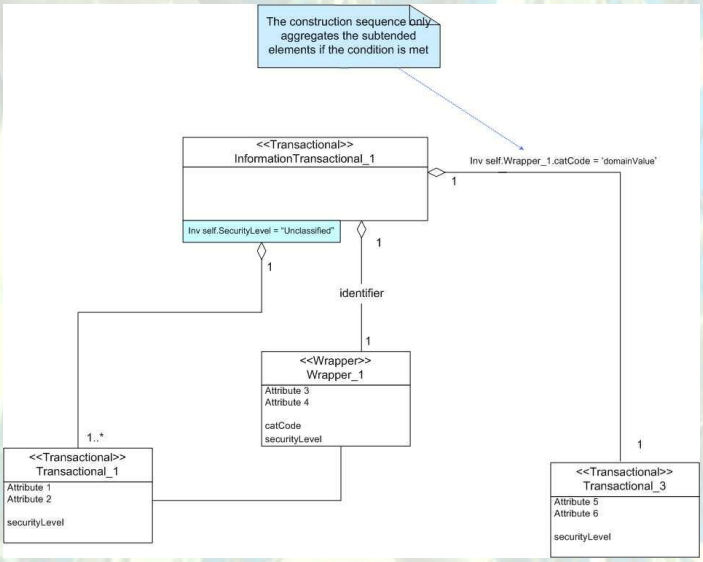

Figure A.8 - Constraints

EntID holds the unique identifier for its corresponding entity in the underlying datastore. entID is associated with classes stereotyped as "Wrapper."

EntName holds the name of its corresponding entity in the underlying datastore. entName is associated with classes stereotyped as "Wrapper."

A.1.7.8 Independent Existence of Information Elements

Each subtended element within a model exists and persists independently from its enclosing classes; as a row of data can exist within a database table without the referential integrity necessary to complete a transaction or build aggregate information sets. This independent existence of information and data elements (a non-traditional interpretation of class diagrams) reflects the reality of information objects:

- Information or data elements can exist in the environment without providing the completeness necessary for meaningful community semantics; and/or
- Information elements may meet the requisite requirements for one community but not another.

A.2 Reading the Models

A.2.1 Semantic Construction

The first principle of the modeling approach is that the mandatory (multiplicities of 1, 1 ..n, 1 ..*) subtended must exist in the information domain as a prerequisite to the continuation of the build of enclosing transactionals. This policy ensures:

- Semantic completeness of all aggregated data sets based on those policies; and
- Semantic completeness of all data marshalled from received information composites (semantics must complete before information and data elements a place in information stores).

A.2.2 Marshalling a Received Data

When information is received its semantic type is identified, the information and data elements are parsed and processed. If all mandatory elements of the semantic are available – the information and data elements can be further processed as appropriate (e.g., placed in a data store).

If the semantic does not complete there are several options to the Designer: from discarding the data and reporting an error to and interactive request (from the producer of the data, from local store, from an operator, etc.) for data needed to complete the semantic. These are operational and design considerations outside the scope of this paper. In general, it is expected that at the COI-level some policy with respect to guaranteeing referential completeness would be established (e.g., a sender can only embed an information reference if it can provide the referenced data on request).

A.2.3 Class Hierarchy

The models can be developed as a top-down model starting with conceptual information sharing agreements which realize the needs of a community or needline. The models can also be developed using a bottom up approach using the data and information attributes of a legacy database application design (database schema) or through and iterative cycles of specification at each of the conceptual, logical and physical layers of the model. The intent is to provide users, analysts and developer an evolutionary method of development that yields reusable architectural components. As previously identified, the modeling constructs, illustrated in Figure A-1, include:

- CONTRACT) - identifies the semantics included in the information sharing agreement.
- SEMANTIC – specifies the information elements to be aggregated in to the document, message or information composite to be shared based on the community agreements.
- TRANSACTIONAL – specify the build-plans of information element from the foundation classes.
- FOUNDATION – identify the based information and data elements and align logical “wrapper” classes that underpin the models. The wrapper classes represent a single instance of an information element in the environment.

The models can also be developed bottom-up and middle-out depending on the use of legacy application and data stores, or the focus of the projects involved. The modeling techniques employed provide the ability to associate class attributes to accommodate differing naming conventions as one moved between the physical, logical, and conceptual (strategic or business) views of the architecture. The techniques also provide the ability to model attribute-method relationships to support data transformations (e.g., when integrating legacy data environments into a system-of-systems environment).

Figure A.1 illustrates the relationships between elements of the models and other views in an architecture framework. It is intended that the policies, or rules, generated from the models realize the elements currently defined by the UPDM; specifically, the realization of a needline’s information-flow and its information-composite (e.g., Message). The information composite is realized by the enforcement or execution of the derived policies by deployed systems, applications and services in the environment. This specification translates the models into multiple platform specific models (PSM), i.e., a set of JAVA Classes (Annex E) and XML Schema (Annex D).

The Contract enables the specification of community semantics outside the constraints of a needline and its associated operational-nodes. The Contract allow the formation on conceptual communities of interests (CoIs) outside the specification of its operational configurations. In its simplest form, a Contract can have a one-to-one relationship with a Needline.

The remainder of this section describes the use of the policy models within an architectural model. These descriptions will reflect a top-down development strategy and are critical to understanding the utility of the SOPES IEDM model. The integration of policy models with the SOPES semantics metamodel enables a rich collection of information management techniques to be executed by an operational information exchange framework.

A.2.4 Stereotypes

The models define the following stereotypes to describe the hierarchy of the models, tying business architecture requirements to the underlying information stores:

- **Contract** identifies a class as a contract;
- **Semantic** identifies a class as a Semantic;
- **Guard** identifies a class as a semantic guard;
- **Transactional** identifies a class an enclosing transactional;
- **Wrapper** identifies a class as a wrapper for an instance of an information element;
- **Entity** (which may be prefixed with the name of the data store and version) identifies a Modeling Element.

A.2.5 Contract

A "Contract" is simply an agreement to exchange information between two or more participants. As illustrated in Figure A.9, the contract is uses to realize the information exchange requirements of either a needline or a community of interest. The Contract sets a policy (exchange rules) by identifying which Semantics are included in the contract. This relationship is established by setting a dependency between the contract and the included semantics.

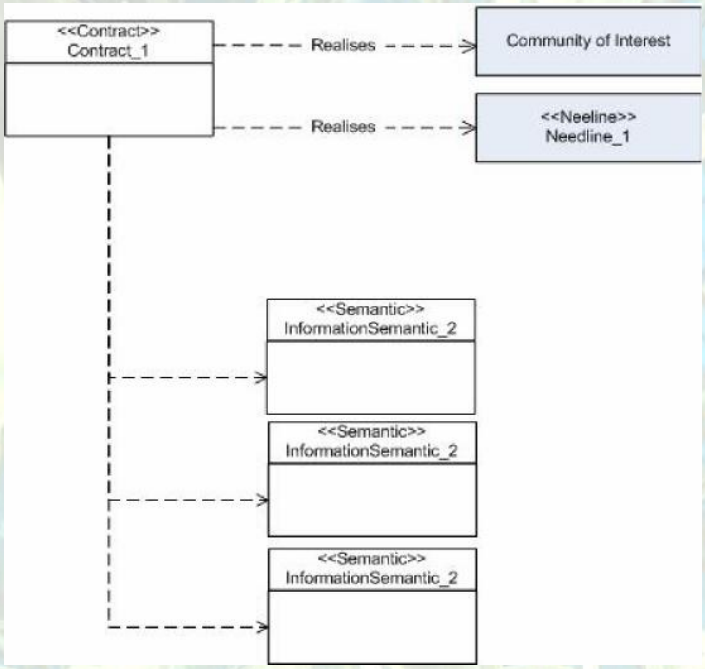

Figure A.9 - Contract

The semantics can be re-used to satisfy the requirements on multiple contracts; meaning, that any time there is a data change to the information contained within a semantic, the change would be reported to all the participants to each of the contracts that include the semantic. Applying this rule means that:

- Semantics become reusable architectural components; and
- Event (data change) driven global update levels of interoperability are achieved.
- In addition to the containment for the semantics comprising and information flow, a "Contract" extends the definition of information exchange requirement to include:
    - The participating operational nodes;
    - The communication channel;
    - The distribution specification (e.g., language, schema, format, syntax and protocol);
    - The agreed quality of service;
    - The potential threats to the exchange;
    - The information sensitivity (e.g., caveat and classification); and
    - The participants and their required accreditations.

These concepts are not core to SOPES IEDM, and thus, are not discussed in this document. They are part of a broader discussion on the role of architecture and architecture frameworks. They play a critical role in specifying the context

and requirements for information exchange and thus are beyond the critical and foundational information (or semantic) interoperability addressed by this specification.

A.2.6 Semantic

A semantic, Figure A.10, represents the specification for a complete data, which is considered meaningful to a community, organization, system or application; meeting one or more of the information flow requirements specification for a needline. The semantic is defined by the community, needline or application interface, while transactionals are closely linked to the underlying data store. The semantic can be thought of as a schema (e.g., IC.XSD) and the InformationComposite thought of as the instance document (e.g., IC.xml).

As illustrated, class attribution is not carried by the semantic. The semantic encloses and carries all attributes contained within its subtended transactionals and wrappers.

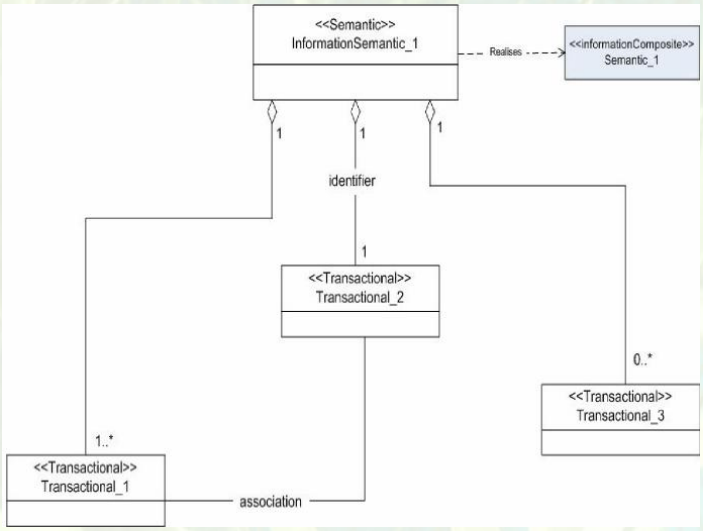

Figure A.10-Semantic

A.2.6.1 Filtered Semantic

A "Filtered Semantic" represents a semantic with all filters, contained in the semantic or its support transactionals, set in the operational environment. Filters can be set by default during design or added during runtime. The constraints, describing a filter are modeled as illustrated in Figure A. 10. A filtered semantic may be used to enforce policies that enable brevity and efficiency by removing from the InformationComposite content (but leaving a reference) that is assumed, or known, to have already been exchanged.

A.2.6.2 Guard Semantic

As "guard semantic" is modeled in the same manner as a semantic (above). In practical terms the "guard" is a semantic stereotyped as - "guard." A Guard builds like a semantic - but works to lock the information contained in its structure from release on any contract. This concept will be explored in later document that address the potential extensions to this UML modeling profile information (semantic) interoperability.

Identifying this concept is intended to illustrate the extensibility of the approach as mandated by the request for proposal (RFP).

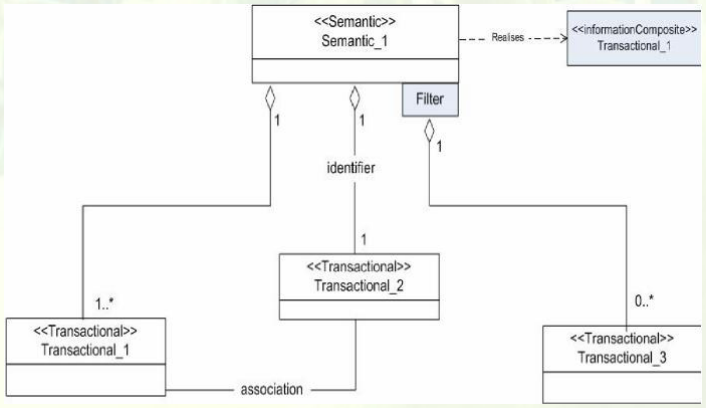

Figure A.11 Filtered Semantic

A.2.7 Transactional

The Transactional, Figure A.12, represents the core concepts within the models. They allow the information architects to construct reusable informational building blocks, upon which to build multiple community semantics. Transactionals document the constructions plans for an information/data store and ultimately link community information needs to the structure of the underlying stores.

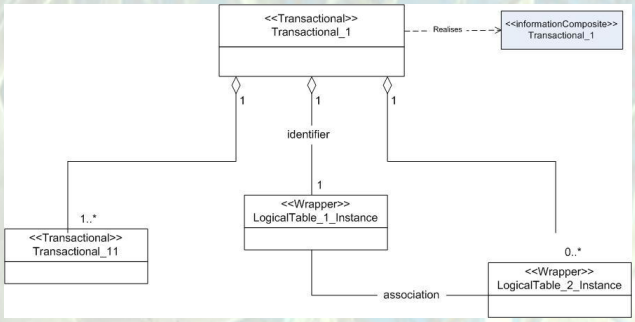

Figure A.12 Transactional

Although there is only one form of transactional, in the course of the follow-on discussion we preface the transaction to identify its role in the construction plan and the structural hierarchy of the semantic:

- Enclosing _Transactional identifies the transactional which forms the focus of the diagram. The enclosing transactional can be identified by the diamonds on the aggregation arc. They are always on the role-end associated with the Enclosing_Transactional.
- Subtended_Transactional (or Support_Transactional) identifies the transactionals contained by a enclosing transactional.
- Identifing_Transactional or Identifying_Wrapper, identifies the subtended transactional or wrapper that carries the keys or identifiers needed to built the aggregation. The Identifing_Transactional or Identifying_Wrapper is identified by the “Identifier” label on its aggregation arc; representing an aggregation arc with a tagged value isIdentifier set to “true.”
- WatchPoint_Transactional is a transactional with an associated watchpoint data event that triggers the build of a semantic and all its subtended transactionals. Each WatchPoint Transactional has one aggregation arc with an 'isWatchPoint' tag set to True.

A single aggregation arc may have both an isIdentifier tag and an isWatchPoint tag. This Transactional must be built starting with the Wrapper at the end of the aggregation arc with the isWatchPoint tag. If both an isWatchPoint and an isIdentifier tag are present in the Transactional model on different arcs then the Transactional may be built as a WatchPoint or a support Transactional.

The rules that identify if a Transactional forms a watchpoint:

- Must contain a composition arc, tagged as 'WatchPoint = True,' that connects to a Wrapper;
- Must hold at least enough data to provide referential integrity when persisted to a data store;
- Complete in it's meaning, ie modeler must include all mandatory as well as optional tables that when combined, provide a coherent picture;
- Comprise basic building blocks for Semantic Artifact; and
- Maybe subtended as well as standalone.

A.2.8 Foundation

A.2.8.1 Wrapper

A Wrapper is a Class that directly maps between the logical data model and the physical data model. Wrappers therefore exist between the Logical Information Security Architecture and the Physical Information Security Architecture. Figure xx illustrates how this mapping is modeled.

A wrapper represents the metadata definition for a single instance (or row in the case of a traditional RDBMS) of data from the underlying information store.

A.2.8.2 Entities

The entity calls represent the physical entity (table) definitions, including the complete attribute metadata definitions, as well as attribute domain and domain business rules. The metadata associated with the entity in assigned to the wrapper and carried through the remainder of the model.

A.2.9 Build Plan

The models specify the policies or rules governing the aggregation or marshalling of information elements included in the community semantics in a manner that is consistent with the structures of the underlying data store. These policies bridge between the community semantics and information patterns (Transactionals) derived for a given data store. The policies (or rules at execution) are enforced by the information applications and services designed to broker information within and between information systems. The models represent a set of architectural views that provide a platform independent specification for the environmental policies governing the exchange of information between operational nodes.

The information pattern provide a systematic build (or construction or navigation) plan for the aggregation of information elements into the Information composite (semantics) agreed to by the community or the participants (systems, applications, services or users) to the exchange. Adherence to the build plan assures that information stores are used in a consist manner and the semantic integrity of the information is maintained.

The SOPES Specification (Annex C) provides guidance on the sequence of the build plan in the form of an "oclConstructionSequence". These elements are intended to be informational in nature and not intended to executable in their current form. It is up to a developer to determine if it is beneficial to include these sequences into the formal expression of function.

A.2.9.1 Initiation of Build

A "Build" refers to the formation of a semantic or transactional within the environment to accommodate a change in information available in the operational environment. The build or processing of new information starts when new information, encompassed by a "watchpoint" transactional, is identified. The existence of a new data event within a watchpoint causes that watchpoint transactional to build, together with all the transactionals and semantics that enclose that watchpoint.

The watchpoint works in a similar manner to a database trigger, but initiates a business object to respond to its environment - rather than a database application triggering a store procedure.

A.2.9.2 Navigation Constraints

Figure A13 illustrates the ability of the architect to adorn the models with constrains that direct specific build steps based on the value of an attribute at runtime. This addresses inclusions or exclusions of specialized objects which can only be determined at runtime. The build plans integrate these navigation constraints.

The SOPES OCL utilizes two constraint patterns that are directly related to the ConstructionSequence. The naming convention for these two patterns include their respective roles within the ConstructionSequence i.e., a_Discriminator_mN0, a_Discriminator_mN 1, or mN0_Enforced_a, mN1_Enforced_a.

These two types of ConstructionSequence constraints are linked by the use of attributes

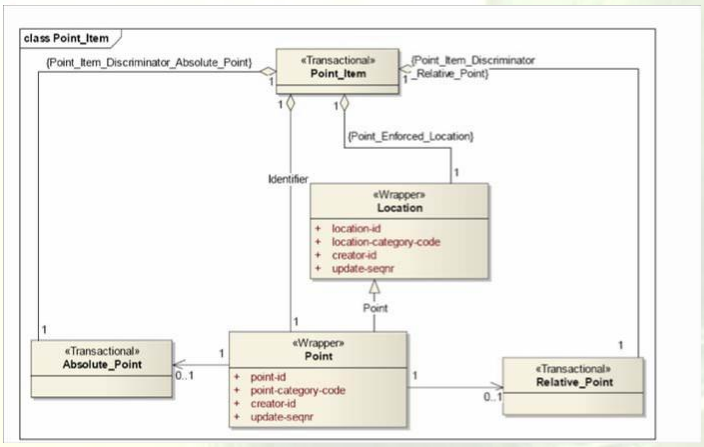

Figure A.13 - Constraining Navigation

whose enumerated domains include a specific reference from the “a” object type to the linked mN# object types.

The use of a_Discriminator_mN# requires that the object "a" have an attribute whose enumerated domain references a set of object types mN# as well as a readPlan between “a” and the object type in the set. For the sake of clarification it should be noted that though this domain explicitly references these mN# object types the enumeration may also includes other choices that are not associated to objects. In the cases of a Wrapper to Transactional based Discriminator the relationship is to the WatchPoint/Identifier of the Transactional

As an example; in Figure A.1 1 objects of type Point have the enumerated domain attribute point-category-code which references objects of type AbsolutePoint the identifier of Absolute_Point (enumerated domain value ‘ABS’) and RelativePoint the identifier of Relative_Point (enumerated domain value ‘REL’) as well as other attribute values which do not reference objects of any type. If in Point_Item an object of type Point exists and the data in point-category-code is ‘ABS’ then the associated (and now mandatory) Absolute_Point object must exist for the Point_Item to complete instantiation correctly, furthermore the ‘ABS’ data also prohibits the construction of a Relative_Point object.

As a counterpoint to the a _Discriminator _mN# constraint is the mN# _Enforced _a which requires that if an object of a set mN# exists then the enumerated domain attribute in “a” must be that value which refers to the specific object type member of mN#. Continuing our example from above, Location has the enumerated domain attribute location-categorycode which associates Location with objects of type GeometricVolume, Line, Point, and Surface. In objects of type Point_Item the object of type Point is the identifier and is constructed before the object of type Location, this requires that the enumerated domain value for the attribute location-category-code be ‘PT’ if the value is any other then the Point_Item should not complete instantiation.

Object constraint language is used to constrain navigation (Section 8) on a containment arc to assure the correct aggregation of subtended element in an information construct and to describe the navigation/construction plan (Annex C) derived from the UML. An exemplar for the use of navigation constraints is illustrated in Figure A. 11.

Table A.3 provides an example of the OCL used to constrain navigations:

Constraint	Details
Point_Item_Discriminator_Absolute_Point	**inv: self.Point.point-category-code='ABS'**
Point_Item_Discriminator _Relative_Point	**inv: self.Point.point-category-code='REL'**
Point_Enforced_Location	**inv: self. Location.location-category-code='PT'**

The Wrapper containing the evaluation attribute (e.g., Point) on a navigation constraint must have a multiplicity of 1 (1..1) within the data pattern. . In reference to Figure A.12 and Table A.3 both Point and Location are constraint evaluation Wrapper instances and thus are required to have a cardinality of 1..1. It should be noted that the Wrapper Attribute which holds the value must comply with the model's domain constraints and as such it is possible that the value held by the Wrapper Attribute could properly be expressed as NULL.

A.3 Developing the Information Exchange Models A.3.1 Top-Down Approach

For new environments, containing no legacy, a top down approach can be used. A top-down approach follows the traditional enterprise architecture methodology.

A.3.2 Bottom-up Approach

In most environments, new architectures are developed based on legacy environments, which have significant investment in their underlying information store. It would be unreasonable to propose a strategy the only supports a top-down approach that may not align with legacy environments in the long run; requiring major upgrade and modification to working systems. A bottom-up approach addresses the possible integration of legacy information applications and stores.

Bottom-up was the approach used to develop the SOPES IEDM specification and prototype implementations. The JC3IEDM, in the form of the MIP information Resource Dictionary (MIRD), was used for the development of the SOPES IEDM Specification Foundation Model. The MIRD content documents the results of a legacy ongoing COI consensus process that captures many unique consultation, command and control requirements in a normalized and generic model. In the process, visibility of the individual business requirements has been somewhat lost.

Today a bottom-up approach can document the available normalized and generic transactional semantics supported by the information stores, but not the business requirements from evolving information architectures.

A.3.3 Hibrid Approach

The most likely approach to be adopted by projects is a hybrid environment where team members would define business requirements in a top-down approach, and others would build up project information based on a bottom-up documentation effort on the legacy information stores. This requires the two efforts to align at the semantic level of the model. The models cater directly to this requirement.

The hierarchical modeling approach provides SOPES implementer with several options and locations where this integration can occur.

A.4 Linking to UPDM

A.4.1 Connecting to the Architecture

One of the first steps in the architectural process is the identification of the needlines between operational nodes. As illustrated in Figure A.14 the needline represents a stereotyped association between two operational nodes.

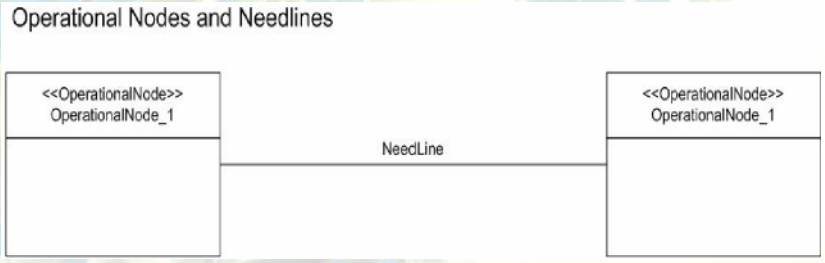

Figure A.14 – Needline

The next step in the architecture process is the identification of the information flows between the operational nodes; bidirectional information flows are illustrated in Figure A. 15. In the runtime environment the information flows realize the exchange of information-composites comprising the content of a community semantic.

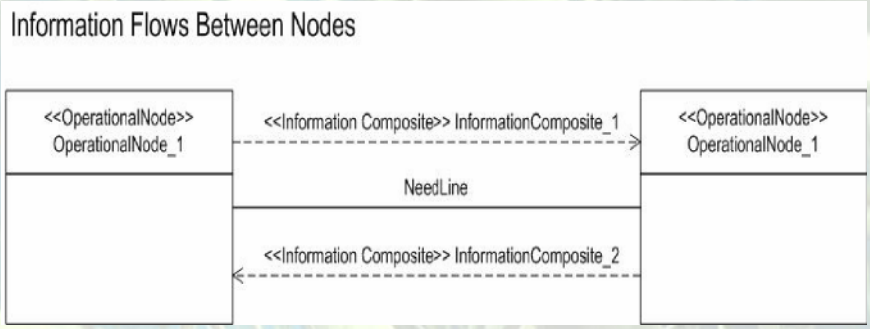

Figure A.15 - Information Flows

The information-composites are realized by the execution or enforcement of the information exchange policies defined by the models. The information exchange policies defined by the models (when executed) realize the information- composites realized by the information flows Figure A.16).

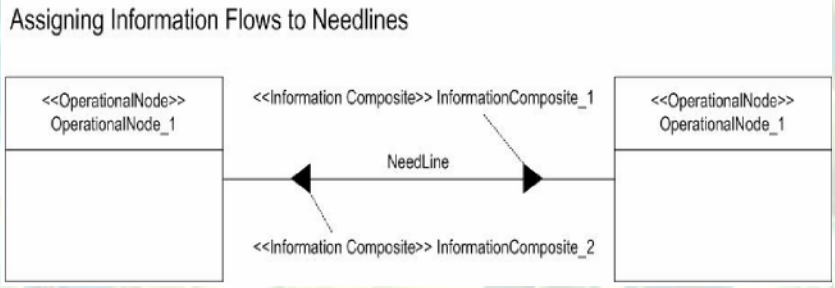

Figure A.16 - Realizing Information Flows

Lastly, the information flows (information composites) are realized on the needlines (Figure A.17). At this point, having semantic realized by the execution or the enforcement of policies derived from the semantic model we have full traceability from the needline to the instance data in the SV-1 1 data store.

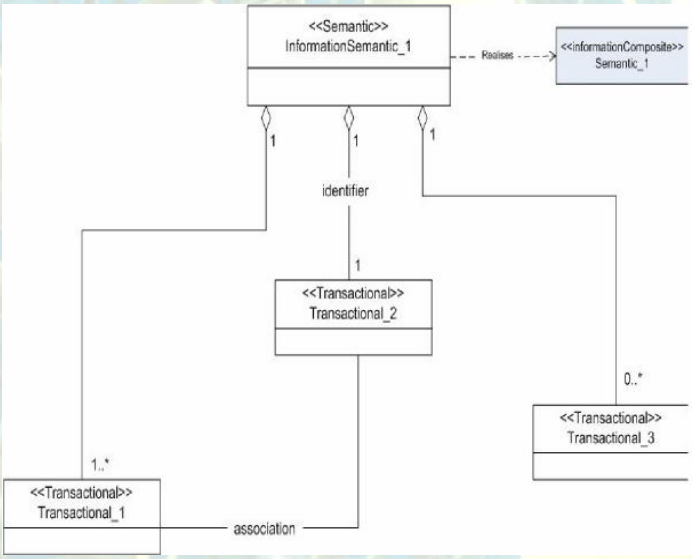

Figure A.17 – Semantics

A.5 Community Defined Completeness

The modeling concepts presented support COIs implementing different approaches to specifying semantic and referential completeness. This section will discuss exemplars, from the SOPES IEDM Specification, showing alternative COI approaches for specifying completeness using the model.

In the SOPES IEDM specification the “Materiel_Item” and its necessary encyclopedic Type data (“Materiel_Type”) are not formally associated until the Semantic (“Materiel _SA”) is formed. In that exemplar, the Materiel_Type is identified as optional (multiplicity: 0..*). For the MIP community this appeared to depart from the referential integrity prescribed by the JC3IEDM Schema (i.e., all Object_Item must be associated with an Object_Type (multiplicity: 1 ..*). As illustrated in Figure 1.18, the Materiel_Item transactional does not include the Material Type information prescribed by the JC3IEDM Schema. The SOPES submitters took the assumption that type_side data is often preloaded into a data store and not required in each transmission. The authors provided the flexibility in the specification for the individual communities to make these assertions in their specified semantics an or through the use of filters.

Figure A.17 - Realization of Information

As illustrated in Figure A.19, the SOPES IEDM exemplar semantic for Materiel (Materiel_SA) encloses Materiel_Item and Materiel_Type, thus aligning the referential integrity of the models and the JC3IEDM Schema. The multiplicity could be defined to require Materiel_Type infomration.

Alternately the community can extend the core models to include additional transactionals which embed the “Item” and “Type” side data. The SOPES specification leaves this within the purview of the individual communities. This approach is provides flexibility and extensibility to the core specification.

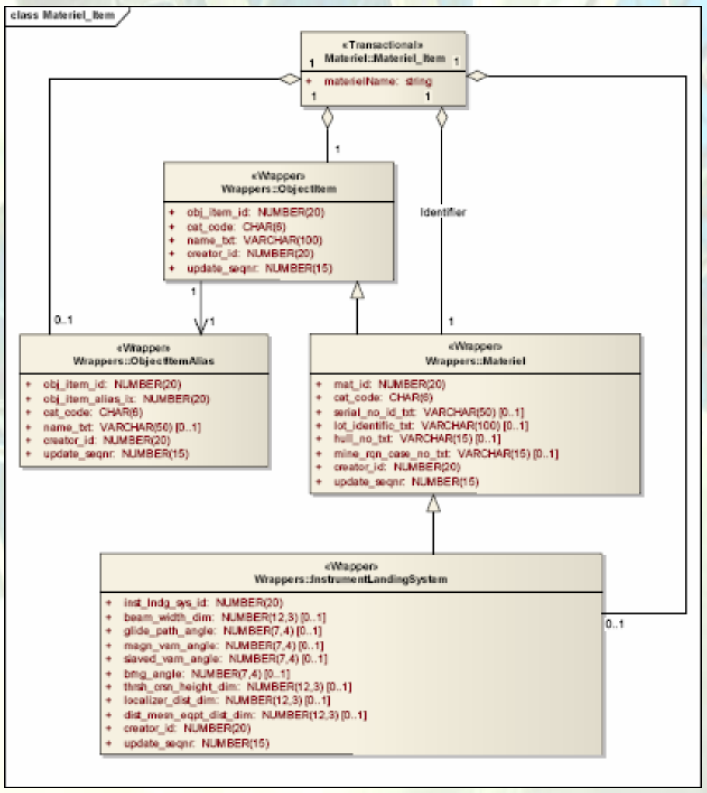

Figure A.82 - Constraining Navigation

A.6 Modeling Extension Examples A.6.1 Domain Filter

Domain Filters can address a number of information quality requirements, including network performance, information overload and priority to name a few. In this section we will look at the use of filters to address elements of security (e.g., Filter based in security tags in the data).

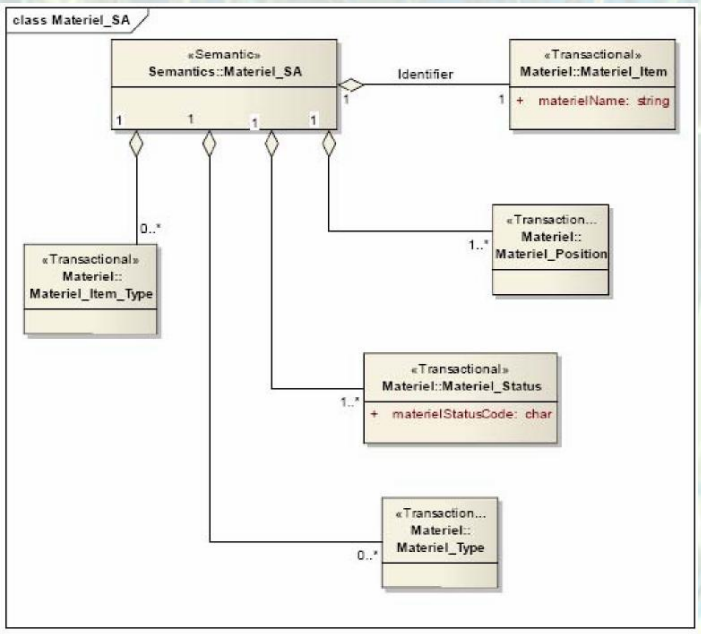

Figure A.18 - Realization of Information

A filter is modeled as a constraint on the role end associated with the enclosing transactional or semantic. Figure A. 19 illustrates the inclusion of a domain filter to the inclusion of “Transactional _1 ,” which limits the inclusion of an information _composite into the build, if its SecurityLevel attribute is set to “unclassified.” An aggregate security attribute can be modeled and evaluated at runtime based on domain specific attribute combinations (i.e., with out the use of the specific subtended “SecurityLevel” attributes). For example, in the context of Action, a domain policy may be that future planned tasks are classified, thus:

- <<invariant>>{ if Wrapper::ActionTask.CategoryCode == "Plan" AND Wrapper::ActionTask.planned-start-datetime > Time. Current THEN Context.SecurityClassificationLevelCode == "Confidential"} (Computed Security Caveat)
- This security example can be extended in to other areas, including:
  - <<invariant>> { if Wrapper::Wrapper_1 .Priority == "High"} (Quality of Service example)
  - <<invariant>> { if Wrapper::Wrapper_1 .PrivacyCode == "Private"} (Privacy example)
  - <<invariant>>{ if Wrapper::Wrapper_1 .Caveat == "NATO" OR Wrapper_1 .Caveat == "5Eyes" } (Release-ability example)

A.6.2 Methods

Within the modeling approach, methods can be used to specify transformations or aggregations of attribute values during the build of an informationComposite. Figure A.21 illustrates the basic construction of a transformation. The attribute(s) associated with the transformation are link through dependency arcs. The index on the arc is a tagged value which indicates the order index of the parameters to the method. The order index identifies the order of the attributes in a method call.

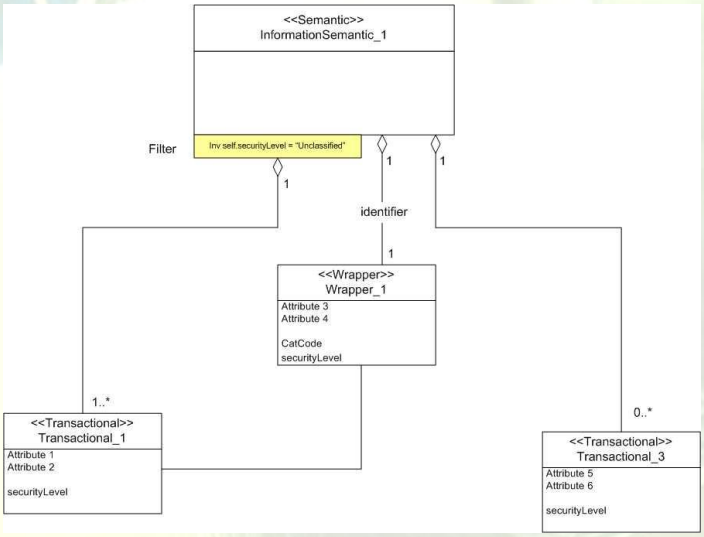

Figure A.20 - Domain Filter

A security example for the use of methods is illustrated in Figure A.22. As illustrated, the method on the enclosing- transactional processes the runtime security level tags of the subtended-elements processes and then sets security level on the generated informationComposite. The diagram does not prescribe the methodology for determining the actual level - merely, that a runtime determination needs to be made.

The specification and design of the method can be handed off to a security analyst or linked to existing policy. Either way, the need for a security decision point in the processing of the information is now captured in the architecture.

Note: the specification of the algorithm can be modeled independently and added to the model. This model only indicates the existence of the decision point.

A.6.3 Forced Method

Figure A.23 illustrates a direct force of the security level base on the existence of the subtended objects by an analyst specification that a combination of subtended information classes automatically mandates a specification of a security level. In this case the classification of the enclosing object is forced to a level independent of the classifications of the subtended objects.

This option adds flexibility to the security considerations of the model.

A.6.4 Selective Replication

Figure A.24 illustrates attribute-to-attribute association. This aspect of the notation can be used to specifically select which data elements (attributes) are processed during the processing of a composite object.

This modeling approach can also be used to migrate from physical names, to logical names to business named (community terminology) as the objects build.

If the attribution is not provided, it is assumes that all attributes are included in the build.

This modeling option also aids in the defining security restrictions on the release of specific data elements.

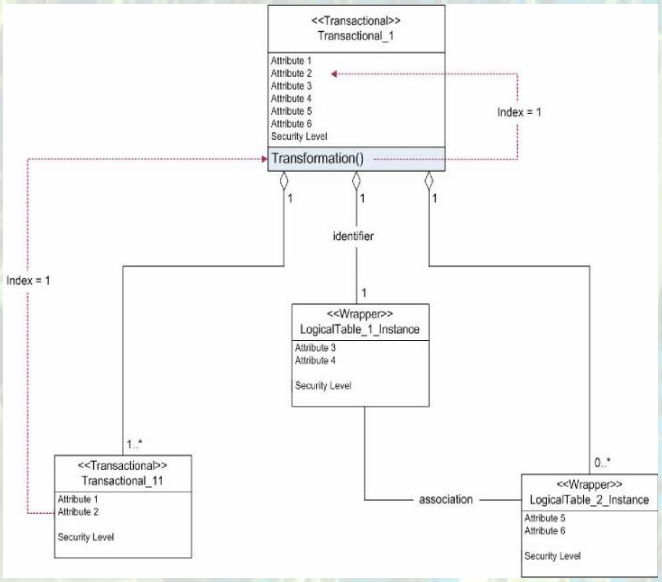

Figure A.21 - Transformation

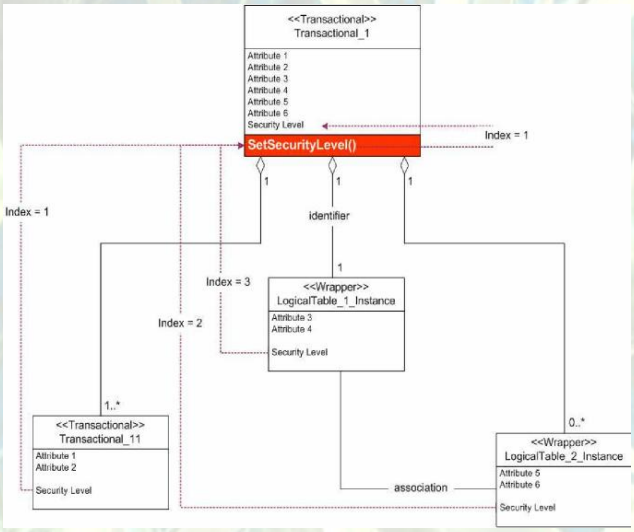

Figure A.22 - Processing Tags and Labels

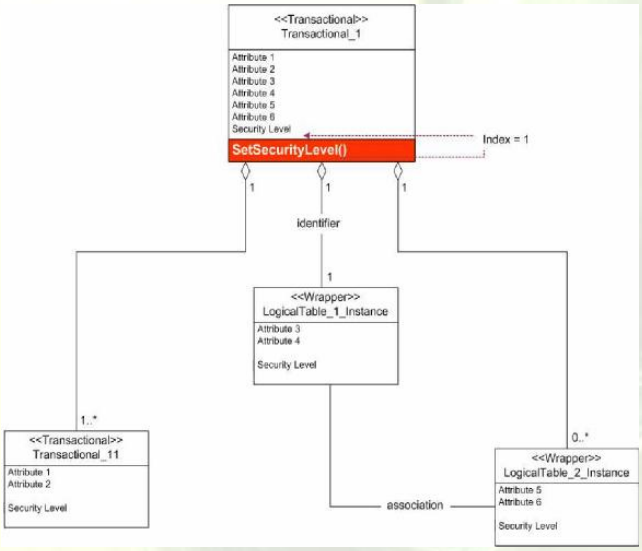

Figure A.22 - Forced Data Change of label

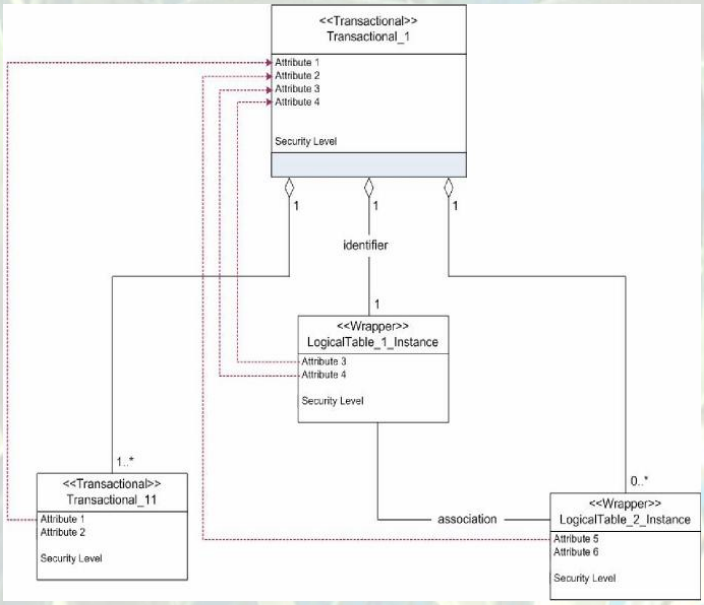

Figure A.23 - Selective Replication

Additional Infromation on SOPES, the modeling profile (including imolemented extensions) and the first SOPES IEDM Implementation (COIL for SOPES IEDM) can be found on the ASMG website (http://www.asmg-ltd.com). Or by contacting:

Mr. Michael (Mike) Abramson, President

Advanced System Management Group Ltd.
265 Carling Avenue, Suite 630
Ottawa, Ontario K1S 2E1
Tel: 613-567-7097 ext 222
Cell: 613-797-8167
Fax: 613-231-2556