Enterprise Change Methodology with MDA

Tony Mallia
Principal Consultant

CIBER Inc. Federal
7900 Westpark Drive, Suite A515
McLean, VA 22102, USA
e-mail amallia@ciber.com

Abstract. This paper describes the practical application of MDA and UML tools in the development of large multi-system projects or system of systems involving multiple development organizations, platforms and tools. A change engineering architectural framework is described with its three view dimensions and how it relates to enterprise architecture. The roles of models at both the change management and methodology views and the separation and use of CIM, PIM and PSM are described in relation to the establishing of integration contracts during the life cycle process. Particular attention is focused on the political reality of multi-organizational development and the delegation of technical decisions. A focus on specifications in the methodology view covers the CIM models (both Ontology and Business Process) and how they transform into PIM Message Templates (Sometimes called a document model) and Component models. Then these PIM models are transformed into PSM component contracts. This paper does not cover PIM and PSM to executable code transformation which is widely covered by current papers. These concepts are illustrated in the implementation of a US Federal Health project which is in operation and in current work being implemented with an XML Schema Factory which shows current off the shelf tools performing transformations.

Introduction

In a large multi-system environment, selection of a single application development tool for all application development is unlikely due to the diversity of language and communications platform technologies and the preference and experience of the various development teams involved.

Successful techniques to produce a coherent implementation across the environment rely on delegation and de-coupling approaches such that the effort can be spread across the teams but that when the parts are assembled together there is high probability of successful integration. Not only will the development be successful but the organization can respond to changes in a routine way maximizing systems development agility.

Change Engineering Architectural Framework

A change engineering framework proposed here has three view dimensions: Perspectives, Focus and Transformation. As shown in figure 1, they provide a space in which to describe the degrees of Transformation:

1. Operational system
2. Change management system
3. Change methodology system
4. Change engineering

These transformation views are applied to the Perspective and Focus dimensions.

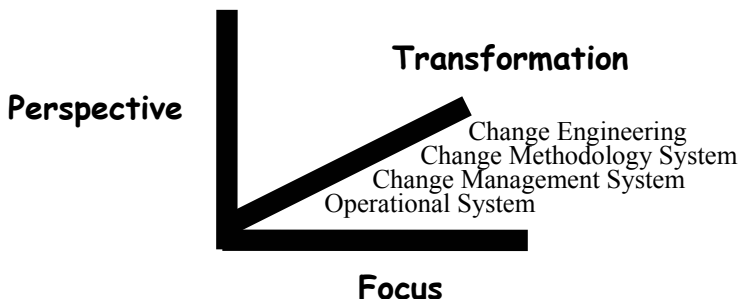


Fig. 1.

Transformation Views

In effect, the organization must look to a Change Management System to make the activities and procedures for change well understood and managed. The Change Management System produces the required Operational System which is used in the Enterprise in day to day activities and is equivalent to the Functioning Enterprise in the Zachman Framework. The development and maintenance of a Change Management System is by a Change Methodology System. MDA provides techniques and tooling to be used by a Change Methodology System to implement the Change Management System.

A system implemented at one transformation view is specified by the model in the higher view. Thus the Operational System is specified by the Change Management Model and the Change Management System is specified by the Change Methodology Model.

Perspective

A number of approaches have defined the perspectives which are targeted towards different players in the organization. A set of 4 perspectives have been found to work in the large multi-system environment. They are shown as the colors in Figure 2.

- The Business perspective defines the environment for the system and contains the manual and computer assisted activities of the operations and their degrees of transformation.
- The Enterprise System Perspective, sometimes called the superordinate system, is the enveloping harness which applies end to end integration around the application systems.
- The Application Systems Perspective, subordinate or subsystems, are the components either bought or built which provide the functionality.
- The Technology Perspective is the language or transport platform on which the application systems and enterprise integration run.

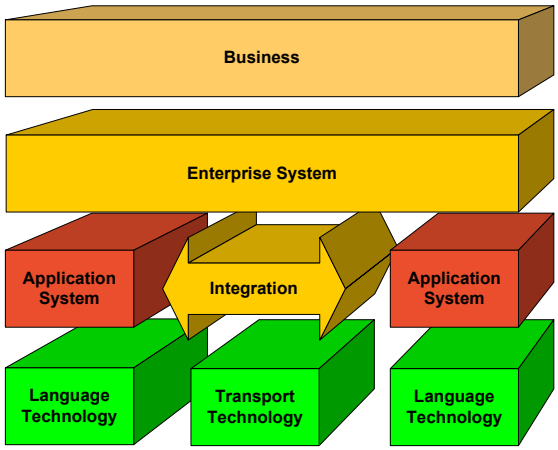


Fig. 2

While these perspectives are not exactly the same as the CIM, PIM, PSM and Platform views of MDA, they provide a better alignment with the organization of the enterprise and the responsibilities for managing large complex systems.

Focus

Focus has been derived from the Zachman Framework but is different in a fundamental way. Where the Zachman framework separates the specification of the system (to be) into the different focus categories, the Enterprise Architecture in this paper defines the focus to be actual instance parts of the system at the appropriate transformation view. The specification is in the model ("What" focus) of the higher transformation view. Thus the Models in the Change Management View describe the 6 focus categories in the Operations View but are not necessarily organized in these categories. The "How" of the Change Management View are the actual activities to produce the Models which describe the Operations View.

The "What" focus defines artifacts or work products which are produced or consumed by the system activities (the "How"). The Where, Who, When and Why define location, participants, schedule and reason for the activities. Thus the Enterprise Change System row describes the project plan and execution of the Enterprise System Change.

Transformation and Focus for Enterprise System Perspective

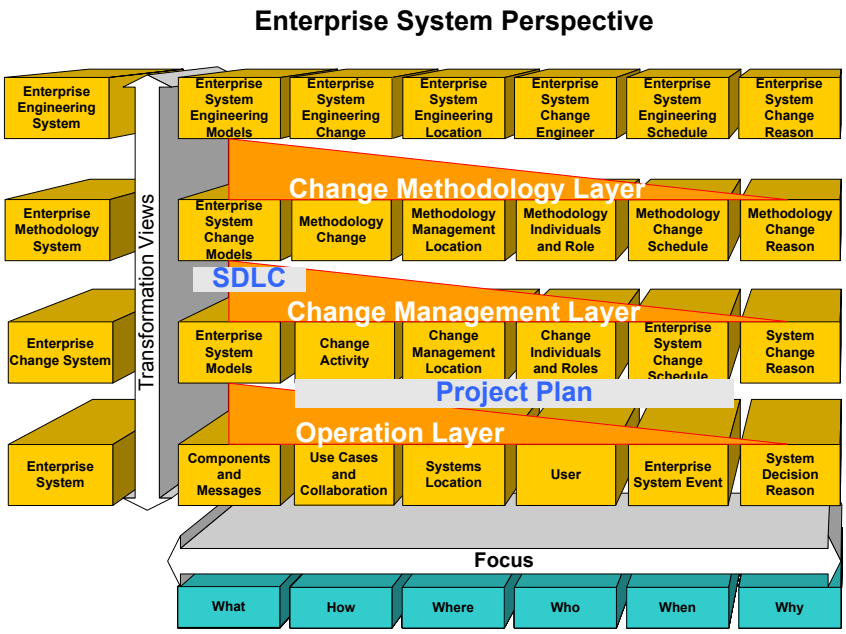


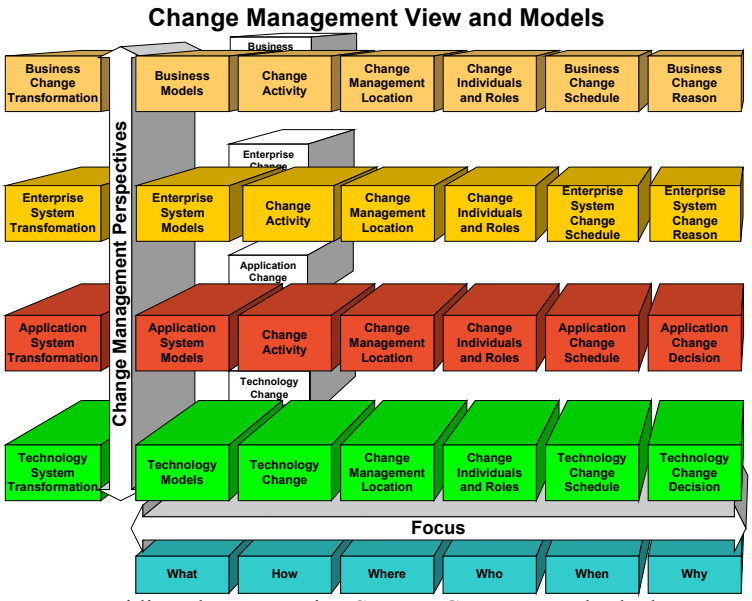
Fig. 3.

Figure 3 extracts a horizontal slice through the Enterprise Architecture for the Enterprise System perspective layer showing the Transformation and Focus dimension Views. The Enterprise Change System is described by the Enterprise System Change Models commonly known as Software Development Life Cycle (SDLC) for the Enterprise System.

Execution of the Change Methodology System results in a definition and deployment of the Enterprise Change System and execution of the Enterprise Change Management System results in the Enterprise System Models and the implementation of the required Enterprise System.

Models of the SDLC can be defined in UML using techniques out of the Systems Engineering Models such as the UML SPEM Profile. The Change Management System is the actual instances and actions of the change process which results in the operational system. Thus a horizontal line of cells represents a system realization.

MDA and Change Management



By providing the Enterprise System Contracts at both the PIM and PSM levels to the Application Systems developers, integration can be facilitated. Some of the PIM model instances from the Enterprise System can be reused in the

The roles of models at the change management system and the coherence of CIM, PIM and PSM allow the bridging between the perspectives. This allows coexistence between MDA and Component Based Architecture where the contracts between Application Systems must be taken to the PSM level to ensure interoperability without platform bridging.

In the Change Management View which is now a vertical slice of the Architecture Framework as shown in Figure 4, model instances of the systems at each perspective are so far disconnected. Coherence between the Business models, Enterprise System Models and Applications System Models would ensure that there is alignment between the Business and the systems implementation and between the Application Systems and the Enterprise Integration Contracts.

Fig. 4.

Application Systems development and some examples to show the models which are reusable.

Although the details of the organization are not discussed here, it is assumed that there is a central architecture group which is able to develop and govern the Enterprise System and its models. Distributed Applications Systems development groups would work with the Enterprise Systems development group to facilitate reuse of models and negotiate integration contracts.

In the same way, Technology Systems need to be selected and integrated in a coordinated fashion whether single technologies are selected or multiple technologies must be bridged. It has been found that it is not necessary to generate code in the Application Systems to provide a high degree of Application Systems independence from the Transport Technology rather a binding layer of the Enterprise System can provide interfaces to the Application System where the nature of the Transport Technology is transparent.

Using MDA in the Methodology Model

The MDA CIM, PIM and PSM model types are aligned to the Business, Enterprise and Application System perspectives as shown in Figure 5. The CIM maps to the Business perspective and the Models contain concepts which exist without a computer system. The PIM maps to both the Enterprise and Application System Models as does the PSM. Since the focus of this paper is the Enterprise System Methodology and the development of the integration contracts to allow successful collaboration between Application Systems, the role of the Enterprise System Models used in development will be explored.

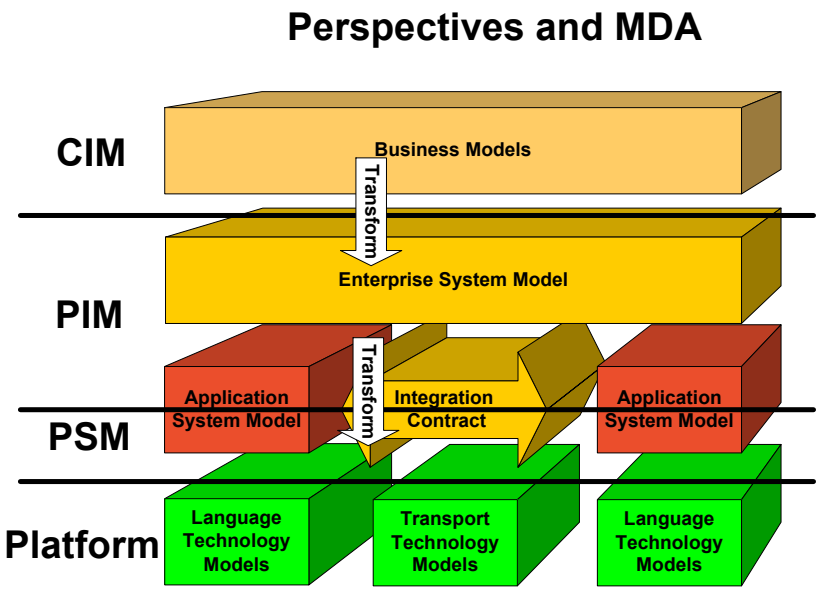


Fig. 5.

The CIM model type as defined in the Methodology System Model is separated into an Information View and a Behavioral View. Both can be defined in UML. Table 1 shows the models involved in the Enterprise System Methodology.

	Information View (What in Operations)	**Behavioral View (How in Operations)**
CIM Business	Business Domain Model (Ontology)	Business Process Model
PIM Enterprise	Message Template Model	Component Model and Collaborations
PSM Enterprise	Message Payload Schema (e.g. XML)	Component Interfaces and Methods

Table 1.

The CIM Information View Domain Model defines the concepts and relationships of the Business. In this sense it is a lower level ontology and can be governed by a middle level ontology as a UML Profile. An example of a CIM Domain Profile (also in the Methodology System Models) might contain Entity, Role, Act and Identity stereotypes. These stereotypes can be used to mark classes which can be use to transform the Domain model to the Component Model – an Act class might generate a "Process Component". However no tool has been investigated which can do this but it might be possible with user defined transformation languages to achieve this.

Typically the Domain model will contain packages for Subject areas as well as Datatypes and Terminologies. The Domain Model requires careful construction because elements will find their way transformed to the PSM of the Integration Contract. The contents of the model are the concepts that exist in the business which are independent of the computers systems.

The Business Process model contains Activities and Object Flows representing the actions caused by business events. Some of these activities can be Enterprise Systems Use Cases where an actor is interacting with the external boundary to initiate or respond to an Enterprise System event. The effect of the system on the business environment can be modeled and a superficial message identification as an Object Flow can be defined. The more fine grained actions in the Use Case are added when showing the interaction of the actor to the system boundary and their linkage to the Component Model Collaborations.

At the Enterprise PIM level again the models are separated into an Information View Message Template model and a Behavioral View Component Model.

The Message Template model represents the payloads of messages being exchanged over the integration transport. A number of techniques have been tried to represent the structure and scope of the payload and the most effective has been found to be a UML class diagram showing a message root class associated with the first content class from the CIM Domain model and limiting the scope through the visibility of elements in the diagram. If the element is not visible then it will not be in the scope of the message.

The Component Model shows both the subsystems which will collaborate along with the collaborations which will realize the Use Cases on the system boundary. Since the Enterprise System is superordinate, it provides the behavioral roadmap for the subsystems interactions. Subsystems are considered as black boxes with external interfaces and behavior – their internal structure or behavior is hidden.

At the Enterprise PSM level the Information View Message Payload is defined in the transport platform's language such as an XML Schema and the Behavioral View is expressed in UML as the specific interfaces and methods which will be used such as Home and Remote Interfaces in the J2EE platform.

Model Transformations

Transformations discussed here in this example Methodology include Domain models and how they transform into PIM Message Templates (Sometimes called a document model) and how the Message Templates transform into the Message Schemas. This paper does not cover PIM and PSM to executable code transformation which is widely covered by current papers.

Business Domain to Message Template Transformation

The transformation from Business Domain to Message template is a selection and restriction process which is performed by hand in the UML tool. The restriction is that any concept or relationship introduced in the message template must have existed in the Domain model. No new concepts other than the type of message can be introduced in the Message template and all structures must be referenced in a Domain package.

Message Template Model to Message Payload

A number of ways of performing the transformation from the Message template to the Payload schema have been tried which include custom scripts to process the content of the model including the generation of CORBA IDL and dictionary descriptions to feed into message transformation bridges.

A commercial tool has been used to convert marked UML Enterprise Message Template and Domain models into XML schemas. This will be described more fully in the XML Schema factory example.

CIM Instance Example

These concepts are illustrated in the implementation of the US Federal Health Information Exchange project which is in operation. The project involves an integration server which exchanges health records between two US Federal Agencies. The records are normalized into standard structures controlled by Message Templates which are derived from the Business Domain model.

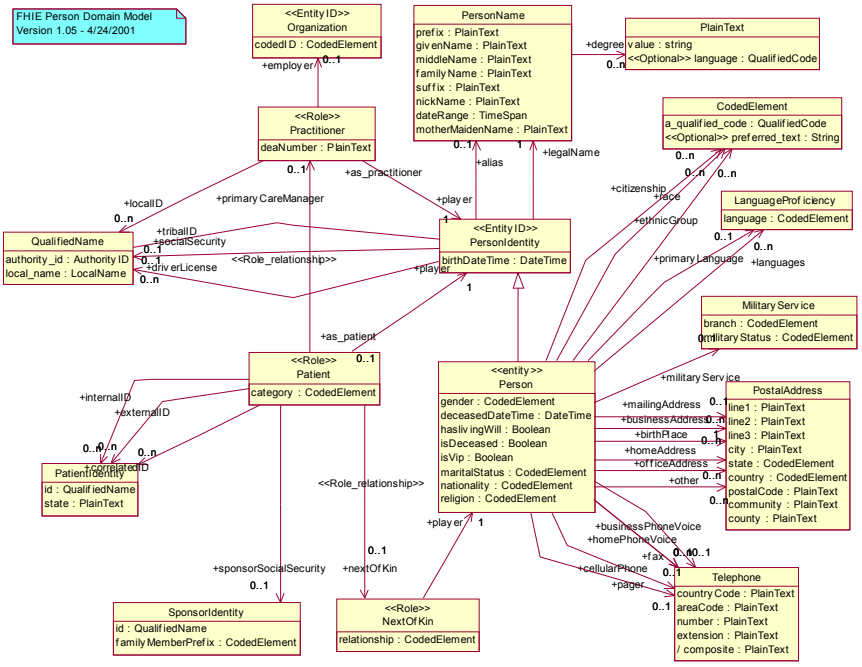


Fig. 6.

An example of a Domain package is shown in Figure 6. The package contains concepts about Person and their roles as Patient and Practitioner as required by the scope of the project.

The relationships show the semantic paths which are permitted.

Message Template

The Message Template example in Figure 7 shows a fragment of the Patient Encounter message template where the diagram includes only the classes and relationships which are included in the message. The Message root is at the lower left of the figure and is associated with a single instance of PatientEncounter class. The Patient Encounter can have an appointment, admission, discharge and procedures. If you walk all the semantic paths from the message root you get all the semantic concepts which can be included in the message.

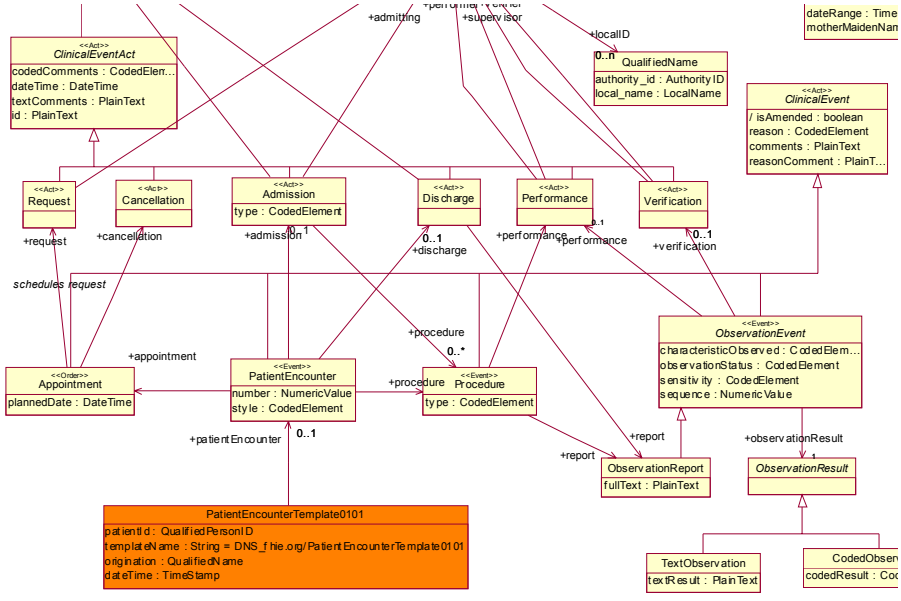


Fig. 7.

In this project the platform was Java and the transport uses serialized Java objects as graphs to convey the PatientEncounter message and the model is used to generate the well formed graph at run time. In this case the run-time bridge reads the model to understand how to construct the graph and transformation is therefore by interpretation.

XML Schema Factory

The second example is current work being implemented with an XML Schema Factory which uses commercial off the shelf tools performing transformations. Figure 8 shows part of the life cycle with the UML editor on the left, the Transformer tool in the middle and the target middleware IDE on the right.

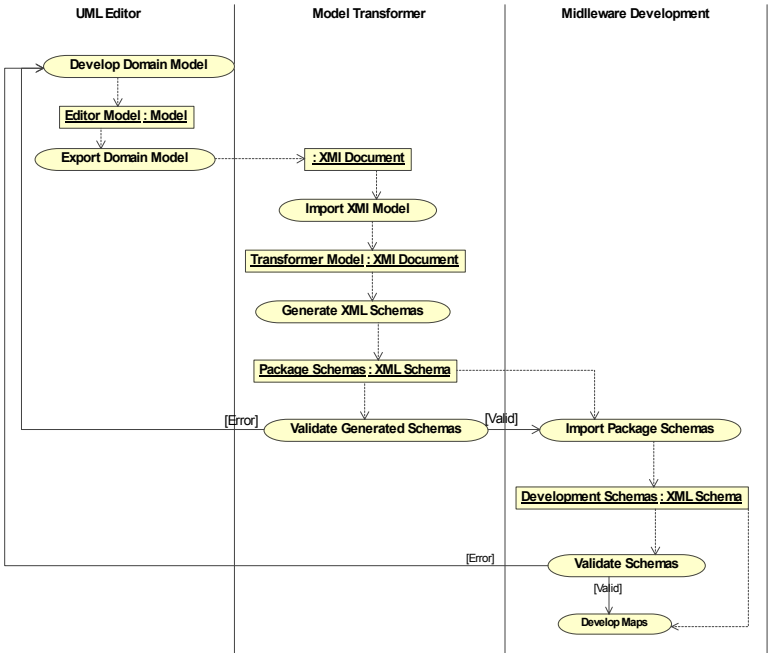


Fig. 8.

The UML editor exports the Domain and Message template models together as a single XMI document which is imported into the Transformer tool. The Transformer tool has preset defaults but can read the marked elements to condition the transformation.

The XML schemas corresponding to packages are generated along with all their external references, namespaces and include statements and can be validated.

They are then exported into the middleware development tool which can use them as the backbone schemas for mapping against other incoming or outgoing schemas and can generate sample documents. The total round trip time is less than a minute.

Figure 9 shows a fragment of the Domain model – in this case part of the Datatypes package and illustrates some simple problems such as defining predetermined string lengths and a Number Datatype which will be generated as a restriction derivation of the XML decimal. The marks appear as stereotypes on classes and attributes as well as a few tagged values.

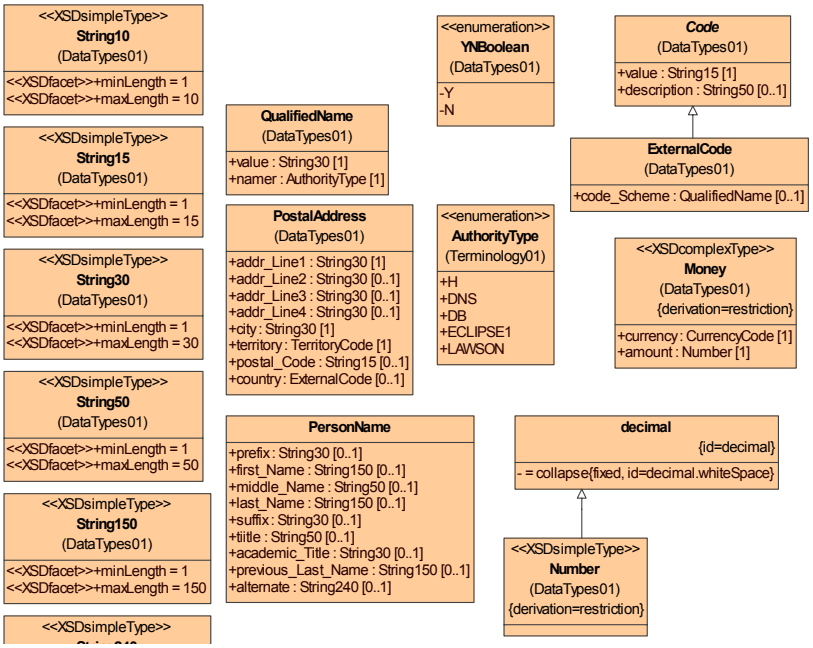


Fig. 9.

The style of schemas produced have very high re-use of common elements and cannot determine exactly the scope for an individual message. The Message template diagram must be used in conjunction with the XML schemas for the Application System developer to understand the payload. Some investigation is continuing into the possibility of the tool developing XML schemas based on the Message Template diagram itself. However this will have to wait for standard diagram exchange in XMI to be implemented by the tool vendors.

The XML factory is going into its first production project and has already demonstrated the strength to build well formed XML schemas and apply the governance needed for successful Enterprise integration. Notice in the generated sample below that all the documentation from the domain model is carried into the XML schemas.

```
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!-- Class: PersonName  -->
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
          <xs:element name="personName" type="dt01:PersonName"/>
          <xs:complexType name="PersonName">
           <xs:sequence>
            <xs:element name="prefix" type="dt01:String30" minOccurs="0">
             <xs:annotation>
              <xs:documentation>Salutary introduction, such as Mr. or Herr
              </xs:documentation>
             </xs:annotation>
            </xs:element>
            <xs:element name="first_Name" type="dt01:String150" minOccurs="0">
             <xs:annotation>
              <xs:documentation>First name of the person
              </xs:documentation>
             </xs:annotation>
            </xs:element>
```

Conclusion

Implementation experience has shown the need to clearly define an Architectural Framework which must be aligned with the development process and organizational structure within an enterprise. A new Change Management Architectural Framework has been explored which allows the allocation of Systems Change to various teams and where MDA and transformation or alignment between models provides a coherence between views whether they are between the Business and the Implemented systems or between systems implemented with contracts in a Component Based Architecture.

This framework declares the relationship between the methodology system and the change system which it defines.