TMF White Paper on NGOSS and MDA

Version 1.0

Authors:

John Strassner, john.strassner@intelliden.com

Joel Fleck, joel.fleck@hp.com

Jenny Huang, jsh@att.com

Cliff Faurer, cfaurer@tmforum.com

Tony Richardson , tonyr@tmforum.com

(c) 2004 The TeleManagement Forum

Document History

The following is a version history of this document:

Version	Author	Date	Purpose
0.0	J. Huang	Jan. 2003	Initial input to NGOSS Architecture team
0.1	J. Fleck	April 2003	Initial thoughts from Architecture team
0.2	J. Strassner	May 2003	First draft
0.3	J. Strassner	July 2003	Additions to NGOSS section
0.4	J. Strassner	August 2003	Additions to MDA and OSS sections
0.5	J. Huang	Sept. 2003	Revision for OMG Boston meeting presentation
0.6	J. Strassner	Sept 2003	Major comments and suggested revisions
0.7	C. Faurer	Sept 2003	Revised Overview and Conclusion
0.8	J. Huang	Oct. 2003	Revision for TMF and OMG marketing review
1.0	J. Huang	Nov. 2003	Release to TMF Marketing for publication

(c) 2004 The TeleManagement Forum

Table of Contents

1 Introduction 1
1.1 Overview 1
1.2 What is NGOSS? 1
1.3 Similarities Between NGOSS and MDA 2
1.3.1 OMG/MDA 2
1.3.2 TMF/NGOSS 2
1.4 Purpose of the Document 2
1.5 Document Organization 3

2 Key Architecture Artifacts 5
2.1 OMG/MDA 5
2.2 TMF/NGOSS 6
2.2.1 Shared Information and Data Model (SID) 7
2.2.2 Security Model 7
2.2.3 Policy Model 8
2.2.4 Business Process Automation 8
2.2.5 OSS Business Applications 9
2.2.6 Framework Services 9
2.2.7 Basic Mechanisms 11
2.2.8 NGOSS Contract 11

3 Lifecycle Methodology 13
3.1 OMG/MDA 13
3.2 TMF/NGOSS 13
3.2.1 Approach 13
3.2.2 Views, Viewpoints, and Aspects 15

4 Summary 17

5 References 18

(c) 2004 The TeleManagement Forum

Table of Figures

FIGURE 1. MODELING LEVELS 5
FIGURE 2. CONCEPTUAL OVERVIEW OF AN NGOSS FRAMEWORK 7
FIGURE 3. THE FOUR VIEWS OF AN NGOSS SOLUTION 15
FIGURE 4. MAPPING OF RM-ODP VIEWPOINTS TO NGOSS VIEWS 16

(c) 2004 The TeleManagement Forum

1 Introduction

1.1 Overview

The aim of this paper is to describe the similarities between the TMF's NGOSS program and the OMG's MDA initiative, and to explain at a high-level how each could help the Enterprise and Service Provider communities to achieve increased return on technology investment. Both NGOSS and MDA strive to support cost effective system specification and component integration through the definition and use of common business and system language – expressed as analysis and design patterns, business process definitions, shared information models, and mappings for moving from technology neutral models to platform specific implementations. Additionally, service abstraction and contractually specified collaboration is used to support a more rapid inclusion of emerging technology.

Through this paper and associated discussions, it is hoped to promote consensus across the industry on best engineering practice for future OSS/BSS solution development built from industry agreed common information models within distributed computing system frameworks and its associated applications.

1.2 What is NGOSS?

NGOSS (New Generation Operations Software and Systems) is a work program governed by the TeleManagement Forum (TMF). TMF, a non-profit organization, consists of more than 340 member companies around the world. The TMF's vision is "*to be universally recognized as the leader and enabler for automating operational management and business process within the global Communications Industry and related supply chains by advancing the available technologies and solutions*".

The aim of the NGOSS program is to deliver a framework that will help produce New Generation OSS/BSS (Operations Support Systems/Business Support Systems) solutions, and be a repository of documentation, patterns, models and code in support of these developments. In this context, "framework" means an architecture and methodology which can support business, system and implementation views of OSS and BSS component-based solutions. The goal of NGOSS is to facilitate the rapid development of flexible, low cost of ownership, OSS/BSS solutions to meet the business needs of the Internet enabled economy.

NGOSS targets the use of commercial off-the-shelf information technologies, instead of technologies unique to the telecommunications industry, as many legacy management systems have done in the past. This approach significantly reduces costs and improves software reuse and operational flexibility, enabling NGOSS-based systems to support a range of new services and new technology environments more easily. NGOSS emphasizes a service-oriented approach based on integration of well-defined collaboration contracts.

(c) 2004 The TeleManagement Forum

1.3 Similarities Between NGOSS and MDA

1.3.1 OMG/MDA

The OMG's mission is to help computer users solve integration problems by supplying open, vendor-neutral interoperability specifications. The Model Driven Architecture™ (MDA™) is OMG's current strategy in solving integration problems.

The MDA defines an approach to IT system specification that separates the specification of system functionality from the specification of the implementation of that functionality for a specific technology platform. To this end, the MDA defines an architecture for models that provides a set of guidelines for structuring specifications, which are expressed as models.[1]

1.3.2 TMF/NGOSS

The NGOSS architecture is described using technology-neutral constructs. These include concepts taken from RM-ODP as well as extensions to the basic UML metamodel to represent fundamental NGOSS concepts and principles. It does not prescribe a single new technology – rather, it allows for a federation of different technological approaches, each of which offers particular advantages at the business and system levels. An NGOSS solution design specification may be implemented using currently available distributed systems information technologies or technologies that are yet to be defined. Critical to this process is the defining, sharing and reusing of common information and data, as this provides a common vocabulary and understanding of both business and system concepts used for analyzing the structure and behavior of a desired OSS/BSS solution. This is done using the Shared Information and Data (SID) model, which is designed as a set of UML models that cover various views of domain information.

1.4 Purpose of the Document

Both TMF/NGOSS and OMG/MDA is aimed to provide benefits to the business leaders and development communities through:

- Technology neutral architecture, i.e. architecture that is sustainable through technology changes
- Cost effective application integration – e.g. interworking and interoperability through application of standards
- Reduced development time for new applications – e.g. use and re-use of business process and design patterns
- Increased return on technology investments – improved mapping of business need to system capability using a model-driven approach
- Rapid inclusion of emerging technology benefits into their existing systems – e.g., service-oriented approach using contractually specified points of interaction

The purpose of this white paper is to bring industry awareness of this common vision and associated work that has been developing somewhat independently by TMF and OMG. To this vision, OMG has

mostly focused on generic modeling techniques and technology development in PIM to PSM mapping. With the same vision, NGOSS has focused on the business process, business object and basic service framework modeling for the Information Communication and Technology (IC&T) industry, as well as the definition of a distributed service-oriented architecture within which these efforts can be realized. For Service Providers, it is important that all vendor products consider the specifications that are being developed by both organizations, as for service realization, one can not do without the other (i.e., both the business semantics and distributed computing standards are equally important for multi-vendor interoperability and for ease of integration).

We hope through an increased industry awareness of these two sets of standards that are, in principle, complementary to each other, we will be able to increase the impact from both organizations to further advance technology development and adoption in pursuit of improved business operating efficiency.

1.5 Document Organization

This version of the white paper is targeted to OMG members. We have assumed readers already have some level of familiarity with the MDA program. However, high level MDA information is given in the document to provide a context for these readers. This paper first describes the key architectural concerns of the MDA and NGOSS approaches. It then focuses on the principles behind building an NGOSS system and how NGOSS can be used. Then, the lifecycle methodology of both the MDA and the NGOSS approaches are described.

NGOSS Terminology

NGOSS draws from many different approaches and standards. The most important of these are defined in this section. The TMF Glossary is listed as reference[2], though in some cases the definitions below represent refinements to the TMF Glossary.

Architecture. Architecture is the set of significant decisions made regarding the specification, design, implementation, and monitoring of a system and its constituent parts. This description includes defining the set of managed objects that make up the system, the interactions between these managed objects, and the constraints placed on using the functionality of a given managed object. In NGOSS, this takes the form of an *architectural style* that governs how these decisions are implemented for a given system. For example, patterns are often used to guide the definition of managed objects and their collaboration with other managed objects.

Behavior (of an object). This refers to how a given managed object interacts with other managed objects. This interaction can be expressed in many ways (e.g., using a collaboration diagram). The DEN-ng spec, which is one of the models which is federated by the SID, abstracts behavior using the notions of *capabilities, constraints,* and *context*. Capabilities define a normalized view of the functionality of a managed object; constraints specify (among other things) restrictions on using that functionality; context defines a specific interaction for which the capabilities and constraints defined are valid.

Business Component. The software implementation of an autonomous business concept or business process. It consists of all the software artifacts necessary to represent, implement, or deploy a given business concept as an autonomous, reusable element of a larger distributed information system.

Business Activity. The definitions of the principles, processes, operations, constraints, and concerns of business which are used, in conjunction with data and interaction definitions, to form an enterprise model; the smallest conceptual unit of business aware functionality. See *Process business rule*, *Data business rule*.

Component. A component represents the unit of deployment and manageability in the NGOSS technology-neutral architecture. It is a unit of composition with contractually defined interactions and explicitly expressed context dependencies. Context dependencies are specified by stating the required interfaces and the acceptable execution platform(s).

Component Framework. A collection of rules, interfaces, and semantics (defined using contracts) that govern the interaction of components plugged into the framework. A component framework typically enforces some of the more vital rules of interaction by encapsulating the required interaction mechanisms.

Contract. A Contract represents the unit of interoperability in the NGOSS framework. It is more than "just" a collection of interfaces – it also contains the semantics governing the use of those interfaces, as well as pre- and post-conditions that must apply so that the component/set of components/systems enters and is left in a known state.

Data Business Rule. A business rule which defines how the integrity of data, along with its maintenance, is performed.

Information Viewpoint. A viewpoint on a system and its environment that focuses on the semantics of information and information processing.

Process. A Process describes a systematic, sequenced set of functional activities that deliver a specified result. In other words, a Process is a sequence of related activities or tasks required to deliver results or outputs to achieve a stated goal.

Process Business Rule. A business rule that is concerned with business process modeling and implementation, not the corporate information model.

Policy and Policy Based Management. Policy Based Management (PBM) is a methodology for managing systems. More formally, PBM is a methodology that describes how one or more applications can manage one or more systems according to a set of (policy) rules. The application of the rules is governed by the finite state machine that describes how the system is to be managed. Note that in this way, it is possible to achieve true end-to-end control, as opposed to having "just" device- or element-level control.

2 Key Architecture Artifacts

2.1 OMG/MDA

MDA provides the guidelines and standards for describing the architecture of the problem in scope unambiguously. The architecture is described in modeling languages with various levels of abstractions (view points). The general division of the modeling levels can be depicted as following:

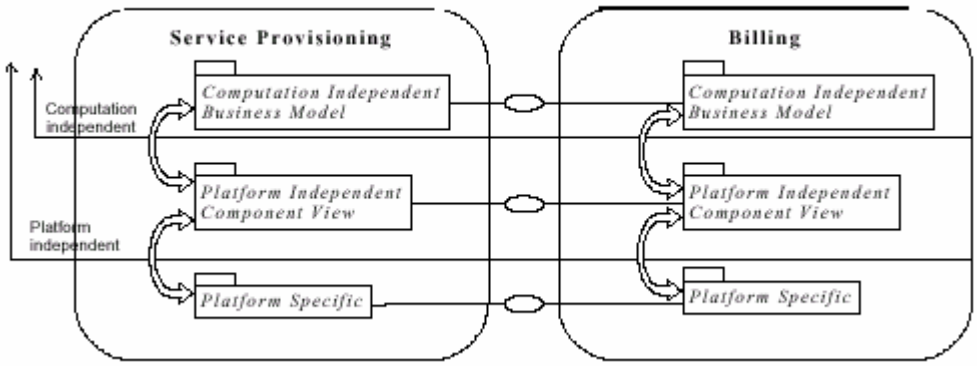

Figure 1. Modeling Levels

The computation independent business model is one in which the Computational details are hidden or as yet undetermined. This could be equated to the enterprise viewpoint of RM-ODP, and roughly corresponds to the business view of NGOSS.

All MDA development projects start with the creation of a Platform Independent Model (PIM), expressed in UML. An MDA model will have multiple levels of PIMs. Although all are independent of any particular platform, each except the base model includes platform-independent aspects of technological behavior. The base PIM expresses *only business functionality and behavio*r. Built by business and modeling experts working together, this model expresses business rules and functionality independent of technology. The clarity of this modeling environment allows business experts to ascertain, much better than they could if working with a technological model or application, that the business functionality embodied in the base PIM is complete and correct. Another benefit: because of its technological independence, the base PIM retains its full value over the years, requiring change only when business conditions mandate.[3]

The use of UML provides the capability to describe Static invariants and pre/post conditions that are particularly important features of an approach to rigorous software engineering called *contract based desig*n. The UML allows formalization of the vocabulary otherwise left imprecise in interface specifications, as an abstract yet precise model of the state of the object providing that interface and of any parameters exchanged. Some current OMG specifications including UML, MOF (Meta-Object

Facility) and CWM (Common Warehouse Metamodel) specifications already use UML and OCL (Object Constraint Language) for specifying constraints.[1]

Once the first iteration of the PIM is completed, it is stored in the MOF and is the input to the mapping step that will produce a Platform-Specific Model (PSM). Specializations and extensions to UML give it the power to express both PIMs and PSMs. Termed a *UML Profil*e, a standardized set of extensions (consisting of *stereotypes* and *tagged value*s) define a UML environment tailored to a particular use, such as modeling for a specific platform. The UML profile for CORBA was standardized by OMG in 2000; other profiles are in process.

During the mapping step, the run-time characteristics and configuration information that are designed into the application model in a general way are converted to the specific forms required by the target middleware platform. Guided by an OMG-standard mapping, automated tools perform as much of this conversion as possible, flagging ambiguous portions for programming staff to resolve by hand. Early versions of the MDA may require considerable hand adjustment here; the amount will decrease as profiles and mappings mature over time.

At the time of this writing, a CIM (Computation Independent Model) is lacking for modeling OSS/BSS applications. A CIM is essential to provide a common business scope and semantics that can be used by all stakeholders, e.g. service providers, software integrators and tool vendors for the next level of interoperable PIM specification. We believe the NGOSS specification is the suitable candidate for this purpose.

2.2 TMF/NGOSS

The NGOSS program consists of a set of business and technical activities that, for the purposes of the OMG focus, are aimed at defining a next generation OSS/BSS, and showing how that next generation OSS/BSS can be implemented. These activities are conceptually equivalent to the OMG/MDA CIM and PIM definitions for a telecommunications UML Profile.

The NGOSS program currently revolves around four main pillars:

1. The architectural pillar is defined in the TMF053[4] series of documents – the NGOSS Technology Neutral Architecture (TNA). These documents define the architectural aspects of a distributed computing system that has NGOSS characteristics.
2. The business process pillar is defined by the Enhanced Telecom Operations Map® (eTOM)[5]. This defines a reference Business Process Framework for identifying and categorizing business activities that a service provider will use.
3. The information pillar is defined by the Shared Information and Data (SID)[6] model series of specifications, which defines managed objects and their behavior and inter-relationships. These models are represented and constructed in a uniform manner using UML and extensions to the UML metamodel.
4. Finally, the compliance pillar defines how components and solutions can qualify as being conformant with a set of NGOSS core and extended principles.

Figure 2 shows conceptually how different parts of the NGOSS Framework fit together.

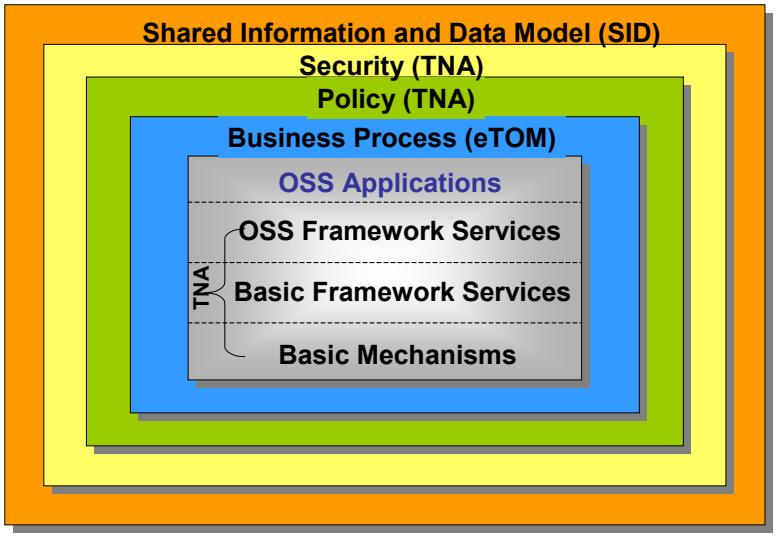

Figure 2. Conceptual Overview of an NGOSS Framework

2.2.1 Shared Information and Data Model (SID)

An NGOSS system is characterized by the usage of a common information model for enabling communication, integration and interoperability. The SID is designed to be more than just a standard representation of data – it also defines semantics and behavior of, and interaction between, managed entities. This set of information models, all provided in a standard representation using standard data types, is used to describe domain information (e.g., customers, orders, network service and configuration definitions) in an NGOSS system.

The SID, expressed in UML, additionally represents information that can be shared and/or reused by NGOSS components. It thus fully represents the information viewpoint of NGOSS. The SID supports business, system, and implementation views thereby enabling different constituencies taking on different roles to concentrate on the set of related information entities of particular interest or concern.

2.2.2 Security Model

An NGOSS system should be designed according to an overarching security model. An implementation of an NGOSS system will require the setup and operation of one or more security mechanisms and policies in order to operate the NGOSS system in a secure manner. To this end, NGOSS uses the security provisions defined by the ISO 17799 Information Security Management standards. This provides a framework to manage and operate an NGOSS system to meet the security

objectives of an operating company. Additionally, the Common Criteria are used as an additional security reference framework[7].

2.2.3 Policy Model

The NGOSS architecture recommends the use of policy mechanisms for defining and managing the behavior of NGOSS elements. Policy-based management is defined as the usage of policy rules to control the state of the system, and managed objects within that system.

NGOSS uses the DEN-ng policy model, which is documented in [8] and [9]. The DEN-ng policy model is a part of the larger DEN-ng model. DEN-ng is an object-oriented information model that has business, system, and implementation aspects. As of this writing, all of the business aspects of DEN-ng have been submitted to the TMF and federated into the SID models. A summary of the theory behind policy management, taken from Reference 9, is as follows:

> *Policy Based Management (PBM) is a methodology for managing systems and components… More formally, PBM is a methodology that describes how one or more applications can manage one or more systems according to a set of (policy) rules. The application of the rules is governed by the finite state machine that describes how the system is to be managed. Note that in this way, we can achieve true end-to-end control, as opposed to having "just" device- or element-level control.*
>
> *Why does PBM use policies? The reason is to be able to control the behavior of a managed system in a predictable and consistent fashion. In order to do this, the characteristics of the system that is being managed must be able to be represented in as much detail as required. Then, policies can be defined that govern each state of the managed object – from creation to deployment to destruction. Without policies, there is no way to coordinate the behavior (e.g., the state and state transitions) of the objects being managed. Furthermore, there is no way to guarantee consistent behavior and reaction to events.*
>
> *How PBM uses policies is critical to the implementation of a PBM system. Many current PBM systems are focused on a particular component in a system, or a set of features, that must be controlled. For example, many Quality of Service (QoS) PBM systems are designed to control a small subset of the features of a device, such as a router. The worry, of course, is the interaction between the QoS features and other features of the router – what if the QoS PBM system makes an adjustment that adversely affects the delivery of some other service or feature that the router is supporting? The answer, of course, is for a PBM system to holistically manage the different components in a system, and the different services that each device supports.*

2.2.4 Business Process Automation

An NGOSS system is characterized by the separation of the hard coded behavior of Components from the software that automates business processes across the Components – i.e., an NGOSS system should be composed of defined services that can be orchestrated using scripting/process management technologies. An NGOSS system is further characterized by externalized descriptions of behavior expressed in a technology-neutral manner.

Process management is the application of modern business management techniques to business processes. This includes techniques for defining, measuring, analyzing, testing and deploying business processes as well as executing them. All of these activities form a part of business management, and so must contribute to the goal of improving business results for telecommunications service providers.

A business process model may invoke lower-level business process models (sub-processes). This means that a business process step that a service provider desires to automate (e.g., verifying that the network can support the provisioning of a desired service) may be made up of one or more lower-level interactions with different system runtime entities that must provide the necessary services; lower-level business process models used in this way must be able to provide one or more Contract instances to which the higher-level business process model can bind. Taking this approach means that *multi-level* process management can be supported.

2.2.5 OSS Business Applications

An OSS Business Application provides services that support the business-related functionality of the NGOSS architecture, such as billing, rating and discounting, network data management, and others. OSS Business Applications deployed in an NGOSS solution are characterized as follows:

2.2.5.1 NGOSS Integrated Applications

NGOSS Integrated Applications are component-based software applications designed from the start for deployment in an NGOSS environment. Applications designed in this way are constructed with the following attributes:

- Functionality accessible through NGOSS Contracts
- Externally visible data conforms to a standard data model that can be mapped onto the NGOSS Shared Information and Data model (SID)
- Business Process and Policy functionality are separated from the implementation of the Application and specified using a set of Business Process and Policy rules. In this manner, Business Process and Policy Management techniques are applied to the separate Business Process and Policy Definitions that in turn orchestrate the flow between components of solutions and applications.

2.2.5.2 Integrated Legacy Applications

Integrated Legacy Applications are software components that were developed outside the scope of an NGOSS architecture which have subsequently been made available as NGOSS components either by:

- Fully encapsulating them with an NGOSS conformant wrapper (i.e., a software wrapper that provides access to the functionality of the legacy application through NGOSS Contracts and maps data made visible by those Contracts onto the common information model used by the NGOSS deployment into which the legacy application is integrated),
- Or, by encapsulating specific functionality of the legacy application by developing adjunct software that provides one or more Contracts and data mapping for that specific service.

2.2.6 Framework Services

These services provide core functionality required to interface an NGOSS component into an NGOSS distributed computing framework implementation. Interfaces to these services are

provided through appropriate contract specifications and can be further split into two sub classes:

2.2.6.1 OSS Framework Services

These Services provide standard OSS/BSS capabilities whose functionality is common to many OSS/BSS services and which can be orchestrated by those OSS/BSS Services. OSS Framework Services are an optional part of the NGOSS architecture, but most likely will be included in some manner in all but the most trivial OSS/BSS deployment. Examples of OSS Framework Services include Logging Services and Auditing Services.

2.2.6.2 Basic Framework Services

These Services provide the capabilities needed to support the patterns of interaction between distributed components that are implementing Business Services. Basic Framework Services will additionally be used by other Services. For example, the process definition for Billing, specified as a group of NGOSS Contracts, would be 'found' by querying and then fetching it from the Repository. The Basic Framework Services are not optional as they are fundamental and necessary to support a distributed computing environment. Each NGOSS deployment must include at least one instance of each of the Distribution Transparency Services that comprise the Basic Framework Services.

2.2.6.2.1 Distribution Transparency Services

The Distribution Transparency Services are defined as NGOSS Technology Neutral Architecture framework level services. These services are fundamental to the construction, deployment and use of solutions in an NGOSS environment. An instance of each Service must be included in each NGOSS deployment. Distribution Transparency Framework Services include:

- **Naming**: to shield prospective users from the complexities (and inherent incomprehensibility) of different types of management information, and to enable defining a federated namespace.
- **Repositories** (or registry services): Store and provide information about available distributed services.
- **Registration Services**: Provide for the administration of services including the addition, modification and removal of services from the Repository, and the ability to browse services previously added to the Repository.
- **Service Location Services**: Facilitate "matchmaking" between potential clients and servers. These Services accept requests for a particular service from potential clients and match that request against registered providers for that service stored in the Repository.

2.2.6.2.2 Federation Model

There are generally two types of methods of federating system services and data: either by providing a communications protocol between services providing similar functions (Service Level Federation), or by providing mechanisms by which the underlying data (both system and user) can be shared (Repository Level Federation). Although both types of federation are recognized by the NGOSS architecture, due to scalability and performance issues introduced with the proliferation of pair-wise Service Level Federations, Repository Level Federation is the (strongly) recommended approach. Repository Level Federation requires an agreement on

a common data model between the federating entities. In addition, since federation is at the Repository level (through a specialized Repository service specification implementing export and import Repository interfaces), federation is achieved transparently to all other Distribution Transparency Services. For example, a client can request location of a Service deployed and advertised in a completely different domain and enterprise using the identical service specification (and without knowing where the Service actually is) that would be used to locate a local Service.

2.2.7 Basic Mechanisms

These Mechanisms provide the baseline functionality which must be implemented in any NGOSS deployment to provide for the inter-working between independently deployed Components and to support the "plug-and-work" paradigm at the most fundamental level.

2.2.7.1 Common Communications Mechanisms

An NGOSS system is characterized by the existence of a communication mechanism (e.g., a messaging bus) or some other form of common communication. All software entities will use one or more communication mechanisms to communicate with each other. Each communication mechanism offers one or more different transport mechanisms. There may be more than one such communication mechanism within a given system implementation, and these mechanisms may represent different technology-specific mappings. The common communication mechanism must provide transport mechanisms consistent with the basic interaction styles defined in the architecture and support any security policies that a service provider has defined for his/her network.

Most importantly, the use of a common communications mechanism enables the standardization of system-wide operations, messages, or events that can be distributed to interested components. This is an important point, as a transport "just" carries information, and by itself doesn't provide the required interoperability to support new generation OSS/BSS solutions.

2.2.7.2 Invocation Mechanisms

These mechanisms provide a common means of performing the steps associated with invoking an operation on a Service instance. There are several steps associated with invoking an operation on a service instance: location of the instance providing the service, proper usage of the communication mechanism, results handling, etc. Security and other policies must be applied at each step in the process. For this reason, Invocation mechanisms must be defined as a convenient way to perform all of these steps and thus ensure the integrity of the operation in the context of the overall system and the client invoking the service.

2.2.8 NGOSS Contract

In addition to the above frameworks and architecture artifacts, the NGOSS contract is the fundamental unit of interoperability. A Contract specifies three things:

- a technology neutral representation of an interface
- the interaction of entities participating in the Contract

- constrained behavior of the entities participating in the specified interaction

Thus, the functionality of a service is made available through a contract-defined interface and one or more services engage in an interaction governed by the behavioral constraints specified in the interaction contract.

NGOSS defines an interface as a named set of operations that characterize the functionality offered by a service. The arguments of an interface, as well as its behavior, are defined using shared information entities to ensure interoperability.

From the above, it can be seen that, at a minimum, a contract is the specification of the interface to a service. However, in order for a contract to be the fundamental unit of interoperability it must specify more than simply an interface, it must also define interaction semantics.

3 Lifecycle Methodology

3.1 OMG/MDA

The life cycle of an application can vary dramatically depending on whether it is being used to build a new application from scratch or just to surgically add a wrapper to an existing application. As is well known, the cost of enhancement and maintenance of an application as well as the cost of integrating new applications with existing applications far exceeds the cost of initial development.

The MDA supports many of the commonly used steps in model driven component based development and deployment. A key aspect of MDA is that it addresses the complete life cycle covering analysis and design, programming (testing, component build or component assembly) and deployment and management. An example is the way in which UML, XMI, MOF and CWM affect the interchange of information between tools and applications. The MDA core is based on OMG technologies (MOF, UML and CWM). These technologies are used to describe PIMs. A PIM can be refined multiple times until the desired system description level is obtained. Then, the infrastructure is taken into account and the PIM is transformed into a PSM. Then, PSMs are refined as many times as needed to achieve the desired level of implementation description.

3.2 TMF/NGOSS

As stated previously, NGOSS encompasses three main pillars – architecture, business process modeling, and the representation of shared information and data – along with a fourth pillar to verify conformance with essential NGOSS principles. In one respect, NGOSS can be viewed as a framework that shows how business processes can be modeled, using shared information, to support a specific architectural implementation that will meet the expressed business need.

Frameworks such as this can be complicated to understand, let alone use. This section will describe the proposed NGOSS Lifecycle Methodology. This is a framework that is currently being developed to guide designers and developers in building NGOSS components.

3.2.1 Approach

The NGOSS Lifecycle Methodology (NLM) is a formalized specification for defining and building NGOSS components. The NLM spans the entire lifecycle of building an NGOSS solution – from the Business Model to the technical design approach to how the solution can be implemented. This is strongly tied to the NGOSS Architecture and draws from four main approaches:

- RM-ODP (the concept of viewpoints, and the modeling of distributed systems and architectures)
- Zachman (an emphasis on Enterprise and Business modeling, and the concept of representing an entity by considering its fundamental aspects (i.e., the “who”, “what”, “when”, “where”, “why”, and “how”)
- MDA (the use of a metamodel, the use of different UML diagrams, the separation of concerns

(CIM-PIM-PSM), and of course the ability to use a model to specify architectural artifacts, and translate modeled entities into working code

- USDP (the concept of an iterative, use-case driven approach)

Figure 3 below depicts the NGOSS Lifecycle. Conceptually, there are four quadrants, representing four aspects of analyzing, designing, implementing and operating an NGOSS Solution. These are termed the business, system, implementation, and run-time views. In the Business View, the eTOM and the SID are used to focus on the concerns of the business: processes, entities and interactions. The eTOM and SID work together to identify business processes and information entities in support of those business processes.

In the System View, the SID, the eTOM, and the NGOSS Architecture are used to focus on system concerns: objects, behavior and computational interactions. Here, the SID System View models are used to add detail to the artifacts identified in the Business View. This in turn enables the business processes to be further refined. Additionally the various business process boundaries are identified and modeled as collaborations in which contracts are used to specify how information and functionality is exchanged between collaborating entities.

The Implementation View uses the NGOSS Architecture and the SID to focus on the componentization of the contracts identified in the System View. Mapping these components to available off-the-shelf solutions is also done in support of the Implementation View.

Finally, the Run-Time View is about actively monitoring the NGOSS system to ensure that the behavior of the operating NGOSS solution is as expected – if not, then it is adjusted appropriately using the NGOSS behavior and control mechanisms: process and policy based management.

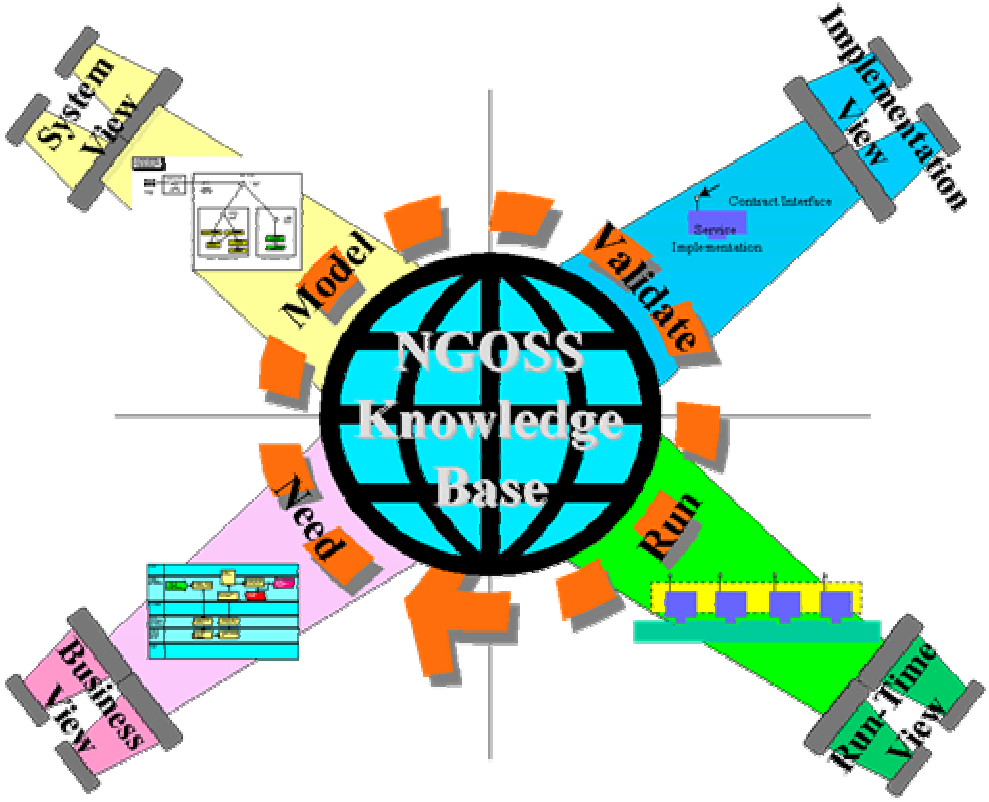

Figure 3. The Four Views of an NGOSS Solution

3.2.2 Views, Viewpoints, and Aspects

Figure 3 above shows the four different "views" of NGOSS. Getting a good 'view' of the NGOSS system under construction is made possible by the well-defined and well-understood conventions used for its construction (the viewpoint language).

> *RM-ODP defines a viewpoint as "…an abstraction that yields a specification of the whole system related to a particular set of concerns." Furthermore, RM-ODP says: "A viewpoint is a subdivision of the specification of a complete system, established to bring together those particular pieces of information relevant to some particular area of concern during the design of the system."*

The key aspects of an NGOSS system are supported by the eTOM, SID, and TNA programs. They combine as shown in Figure 3 to define different views of the NGOSS system as a whole. Figure 4 shows how these different views map onto RM-ODP viewpoints.

Error!

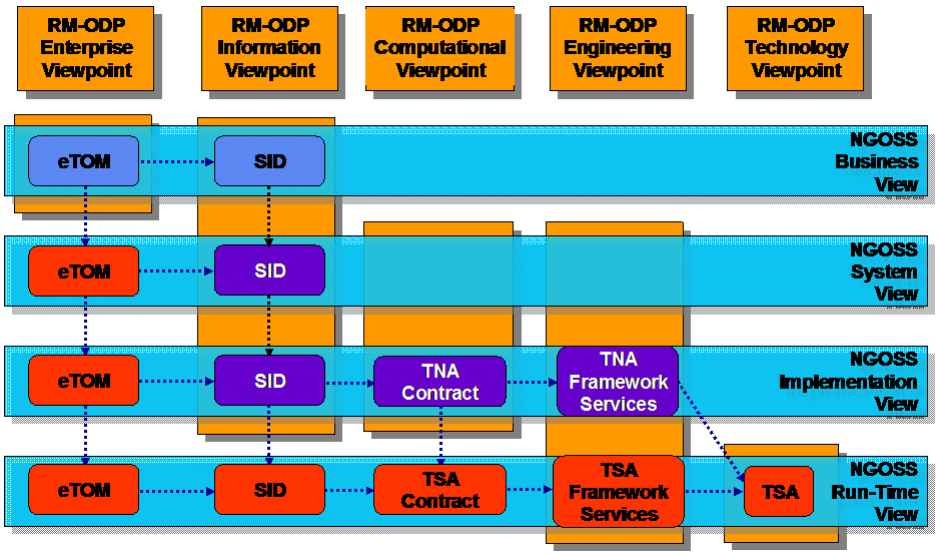

Figure 4. Mapping of RM-ODP Viewpoints to NGOSS Views

Figure 4 above shows that the RM-ODP Viewpoints are centered on a small set of separate core concepts that each focus on the workings of a particular viewpoint. In contrast, the four NGOSS views are defined as encompassing all of the RM-ODP viewpoints. Currently, the SID program spans all four views; the eTOM is in the process of defining support for the system, implementation, and run-time views. The NGOSS Architecture, by definition also encompasses all four views. The NGOSS Technology Specific Architecture (TSA) is responsible for interpreting the NGOSS Architectural Framework and defining the mechanisms and techniques needed to render technology specific solutions such as CORBA, J2EE, WebServices and etc.

4 Summary

This paper grows out of the relationships and similarities between the TMF/NGOSS and OMG/MDA efforts. While following an approach similar to MDA, NGOSS has chosen to focus on building a framework for identifying and specifying well-defined business and system views on a modeled OSS/BSS solution. The OMG has primarily focused on generic modeling techniques and technology development in PIM and PSM mappings. It can now be seen that the work of these two groups is complementary to a common vision of a model-driven approach to technology neutral solution specification and platform specific implementation.

The TMF and OMG industry standard information models and distributed computing system frameworks are on a path to providing a necessary and sufficient reference set for building the next generation of OSS/BSS solutions. The goal now must be to increase industry awareness of this common vision and the associated work that has been progressing within the TMF and OMG and to promote consensus across the industry on adopting these best practices for future OSS/BSS solution specification and development.

5 References

All TMF references can be found at the TMF web site of "Document Solution Suite" within the TMF Knowledge Base: http://www.tmforum.org/browse.asp?catID=1367&sNode=1367&Exp=Y

[1] "Model Driven Architecture - A Technical Perspective", http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01

[2] TMF, "TM Forum Glossary", TMF044, April 2003-08-23

[3] "Developing in OMG's Model Driven Architecture (MDA)" ftp://ftp.omg.org/pub/docs/omg/01-12-01.pdf

[4] TMF, TMF053 and TMF 053x series

[5] TMF, "*GB921: eTOM – the Business Process Framework, version 3.5*", July 2003 (3 Addenda are currently in force)

[6] TMF, "GB922: *Shared Information/Data (SID) Model: Concepts, Principles, and Business Entities*", July 2003 (14 Addenda are currently in force)

[7] TMF, "*The NGOSS Security Principles*", TMF 053s, v0.3 (draft), June 2003

[8] TMF, "*Shared Information/Data (SID) Model – Addendum 1-POL, Common Business Entities Definitions – Policy*", v1.0, July 2003

[9] J. Strassner, *Policy Based Network Management – Solutions for the Next Generation*", Elsevier, ISBN 1-55860-059-1