Terence J Blevins
The Open Group
CIO and VP
44 Montgomery St, Suite 960
San Francisco, CA 94104

John Spencer
The Open Group
Director, Architecture Forum
Thames Tower, 37-45 Station Rd
Reading RG1 1LX, UK

Fred Waskiewicz
OMG
Director of Standards
250 First Ave., Suite 100
Needham, MA 02494, USA

The Open Group and OMG

TOGAF ADM and MDA®

The Power of Synergy

Terence J Blevins
The Open Group
CIO and VP
44 Montgomery St, Suite 960
San Francisco, CA 94104

John Spencer
The Open Group
Director, Architecture Forum
Thames Tower, 37-45 Station Rd
Reading RG1 1LX, UK

Fred Waskiewicz
OMG
Director of Standards
250 First Ave., Suite 100
Needham, MA 02494, USA

TOGAF ADM and MDA

The Power of Synergy

Executive summary

"OMG's Model Driven Architecture (MDA) is a standards-based approach to system development, which increases the power of models in that work. It is model-driven because it provides a means for using models to direct the course of understanding, design, construction, deployment, operation, maintenance and modification." [1] An MDA approach starts with the well-known and long established idea of separating the specification of the business functionality of a system from the details of the way that system uses the capabilities of its underlying platform technology to achieve that functionality. An MDA approach is independent of development methodologies as well as technology. This separation of business functionality from computing technology and methodology preserves a company's core software assets in the constantly changing world of information technology.

TOGAF is a "detailed method and a set of supporting tools - for developing an enterprise architecture." [2] The TOGAF Architecture Development Method (ADM) explains how to derive an organization-specific enterprise architecture that addresses business requirements. The ADM provides a reliable, proven way of developing the architecture; architecture views which enable the architect to ensure that a complex set of requirements are adequately addressed; linkages to practical case studies; and guidelines on tools for architecture development.

A framework provides "a taxonomy for relating the concepts that describe the real world to the concepts that describe an information system and its implementation." [3] One of the best-known examples of a framework is the Zachman Framework. [3]

The relationship between these concepts at the high level is simple:

Pick your favorite framework; use TOGAF to fill it up; and use MDA to empty it!

This is an oversimplification, but in essence it captures the key distinctions. A more detailed statement of the relevant concepts would be as follows:

- Frameworks are used to categorize the information that is needed in order to fully describe an enterprise, and to store that information, typically with the support of an appropriate repository tool;

- The TOGAF ADM is used to develop a description of an enterprise architecture that meets the business needs of the enterprise, populating the framework with appropriate architectural models in the process;

- MDA is strong in software design and implementation that results in implemented enterprise software addressing those business requirements, thereby completing the cycle from requirements to implementation.

Again, this position is a simple one, but one that, if exploited by industry, can realize great benefits. "What if we could paint a picture where customers see better returns from architecture if they use TOGAF's ADM to understand why they are going where and if they use MDA to preserve throughout implementation the linkage to ADM's output and thus to the business rationale for an investment? What if we could show consultancies that they can take on larger projects by easily leveraging efficient focused sub-contractors through open methods? What if we could convince tools vendors that they could plug into a proven chain of methods [and standards] from The Open Group and OMG, without having to cover the whole span from business rationale to running code?" [4]
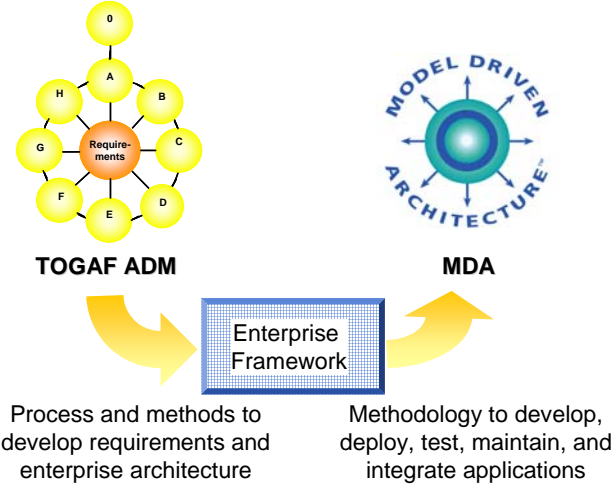

Figure 1: TOGAF ADM / OMG MDA Synergy: Pick – Fill – Empty

The power of this potential synergy is huge.

The remainder of this paper describes TOGAF ADM and MDA, and the synergy that can result from these two key industry resources being used in complementary ways. It introduces the areas where the resources complement each other, and then describes how one can best use the TOGAF ADM and MDA together.

What is TOGAF?

TOGAF is an industry standard architecture development method and resource base that may be used freely by any organization wishing to develop enterprise architecture for use within that organization.

TOGAF has been developed and continuously evolved since the mid-90's by representatives of some of the world's leading IT customer and vendor organizations, working in The Open Group's Architecture Forum. Details of the Forum, and its plans for evolving TOGAF in the current year, are given on the Architecture Forum web site [6].

Two versions of TOGAF are available:

- TOGAF Version 8 ("Enterprise Edition"), first published in December 2002 and republished in updated form as TOGAF Version 8.1 in December 2003

- TOGAF Version 7 ("Technical Edition"), published in December 2001

TOGAF Version 8 uses the same underlying architecture development method that was evolved, with particular focus on Technical Architectures, up to and including TOGAF Version 7. TOGAF Version 8 applies that architecture development method to all the domains of an overall Enterprise Architecture, including Business, Data, and Application Architecture, as well as Technical Architecture.

What Specifically Does TOGAF Contain?

TOGAF provides a common sense, practical, prudent, and effective method of developing enterprise architectures.
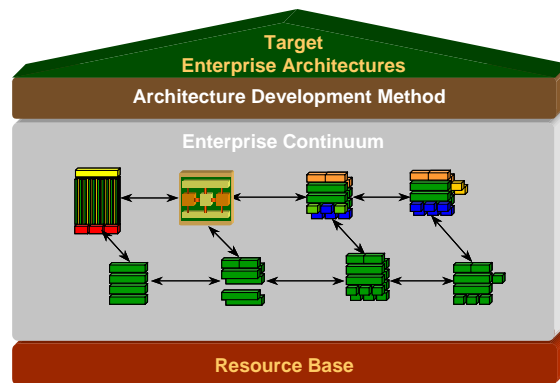

Figure 2: TOGAF

TOGAF consists of three main parts depicted by the Figure 2.

1. The TOGAF **Architecture Development Method** (ADM), depicted in Figure 3, explains how to derive an organization-specific enterprise architecture that addresses business requirements. The ADM provides:

- A reliable, proven way of developing the architecture

- Architecture views which enable the architect to ensure that a complex set of requirements are adequately addressed

- Linkages to practical case studies

- Guidelines on tools for architecture development

It is germane here to point out that in the phases A, B, C, and D depicted in Figure 3, the TOGAF ADM calls out for the creation of models to capture views resulting from the architecture work.
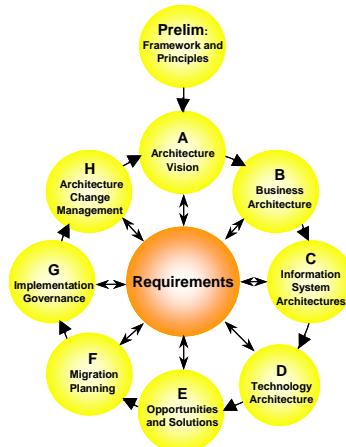
Figure 3: TOGAF ADM

2. The TOGAF **Enterprise Continuum**, depicted in figure 4, is a "virtual repository" of all the architecture assets - models, patterns, architecture descriptions, etc. - that exist both within the enterprise and in the IT industry at large. These are assets that the enterprise considers itself to have available for the development of architectures. At relevant places throughout the TOGAF ADM, there are reminders to consider which architecture assets from the TOGAF Enterprise Continuum the architect should use, if any.
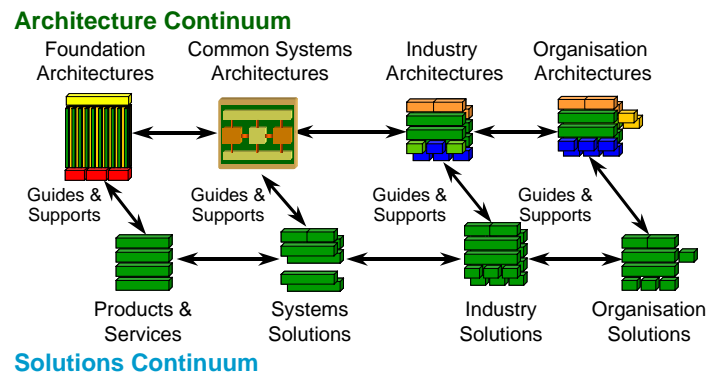

Figure 4: TOGAF Enterprise Continuum

TOGAF itself provides two reference models for consideration for inclusion in an enterprise's own Enterprise Continuum:

The *TOGAF Foundation Architecture* -- an architecture of generic services and functions that provides a foundation on which specific architectures and architectural building blocks can be built. This Foundation Architecture in turn includes:

- The TOGAF Technical Reference Model (TRM), which provides a model and taxonomy of generic platform services; and

- The TOGAF Standards Information Base (SIB), a database of open industry standards that can be used to define the particular services and other components of an enterprise-specific architecture

The *Integrated Information Infrastructure Reference Model*, which is based on the TOGAF Foundation Architecture, and is specifically aimed at helping the design of architectures that enable and support the vision of "Boundaryless Information Flow" [5].

3. The TOGAF **Resource Base**, which is a set of resources - guidelines, templates, background information, etc. - to help the architect in the use of the ADM. The TOGAF Resource Base provides information on the following:

- Architecture Boards

- Architecture Compliance

- Architecture Contracts

- Architecture Governance

- Architecture Maturity Models

- Architecture Patterns

- Architecture Principles

- Architecture Skills Framework

- Architecture Views

- Building Blocks Example

- Business Scenarios

- Case Studies

- Glossary

- Other Architectures / Frameworks and their relationship to TOGAF

- Tools for architecture development

- A mapping of the TOGAF ADM to the Zachman Framework

What is MDA?[1]

MDA Overview and Concepts

"MDA is an approach to system development, which increases the power of models in that work. It is model-driven because it provides a means for using models to direct the course of understanding, design, construction, deployment, operation, maintenance and modification."

An MDA approach starts with the well-known and long established idea of separating the specification of the business functionality of a system from the details of the way that system uses the capabilities of its underlying platform technology to achieve that functionality. MDA derives its three primary goals from this idea: portability, interoperability and reusability through architectural separation of concerns." [1]

An MDA approach is independent of development methodologies as well as technology. This separation of business functionality from computing technology and methodology preserves a company's core software assets in the constantly-changing world of information technology.MDA provides an approach for, and enables tools to be provided for:

---

[1] The authoritative definition of MDA concepts is given in the MDA Guide available from the Object Management Group. This subsection explains some of the key concepts, particularly those that relate to TOGAF and architecture development. Specific quotes and much of the text should be attributed to the MDA Guide [1].

- specifying a system independently of the platform that supports it,

- specifying platforms,

- choosing a particular platform for the system, and

- transforming the system specification into one for a particular platform.

This is achieved through MDA's extensive use of modeling and abstraction.

The following paragraphs summarize key MDA concepts.

Viewpoints

In MDA a viewpoint on a system is a technique for abstraction using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within a system. This definition is very consistent with IEEE 1471. MDA provides modeling languages for three primary modeling viewpoints: a computation independent viewpoint, a platform independent viewpoint, and a platform specific viewpoint.

The Computation Independent Viewpoint and Computation Independent Model (CIM)

The *computation independent viewpoint* focuses on the environment of the system, and the requirements for the system; the details of the structure and processing of the system are hidden or as yet undetermined (this is abstraction.)

A *Computation Independent Model* (CIM) is a representation of a system from the computation-independent viewpoint.

A CIM focuses on the business environment in which a system will be used. The technical details of the structure of the system are hidden or as yet undetermined. The model is oriented to the stakeholders of the system, and may hide much or all information about the use of automated data processing systems or information technology in general. It is useful, not only as an aid to understanding a problem, but also as a source of a shared vocabulary for use in other models, and as an aid to understanding what the system is to do.

A CIM in MDA terms might actually consist of two or more UML models, typically from the ODP enterprise and information viewpoints, some providing more detail than others, or focusing on particular stakeholder concerns.

In OMG terms, this kind of model is also commonly known as a *domain model* or *business model*.

In TOGAF terms, a CIM is referred to as a *business architecture model* or *data architecture model*.

The Platform Independent Viewpoint and Platform Independent Model (PIM)

The *platform independent viewpoint* focuses on the operation of a system while hiding the details necessary for a particular platform. A platform independent viewpoint shows that part (the business functionality) of the complete specification that does not change from one platform to another.

A Platform Independent Model (PIM) is a representation of a system from the platform independent viewpoint. The platform independent viewpoint is computationally complete (executable), uncommitted to any specific platform (i.e., portable), oriented to stakeholders (prototyping) and developers (a testable specification), and constitutes a long-life asset. It

focuses on the operation of a system, while hiding the details necessary for a particular platform.

A platform independent viewpoint shows that part of the complete specification that does not change from one platform to another. A platform independent viewpoint may use a general purpose modeling language, or a language specific to the area in which the system will be used.

In TOGAF terms, a PIM is referred to as an *applications architecture model*.

The Platform Specific Viewpoint and Platform Specific Model (PSM)

A *platform specific viewpoint specific viewpoint* combines the platform independent viewpoint with an additional focus on the detail of the use of a specific platform by a system.

A Platform Specific Model (PSM) is a representation of a system from the platform specific viewpoint. The platform specific model is precise and complete, conforms to the constraints of a specific (class of) platform(s), probably generated from a PIM, and is oriented to technology experts. It combines the platform independent viewpoint with an additional focus on the detail of the use of a specific platform by a system.

In TOGAF terms, a PSM is referred to as a technology architecture model.

OMG domain (vertical market) specifications take the form of normative PIMs expressed in UML, augmented by normative PSMs for at least one target platform MDA also has the concept of a Platform Model.

A Platform Model (PM) provides a set of technical concepts representing the different kinds of parts that make up a platform and the services provided by that platform. It also provides, for use in a PSM, concepts representing the different kinds of elements to be used in specifying the use of the platform by an application. For example, the CORBA Component Model (CCM) provides concepts such as EntityComponent, SessionComponent, ProcessComponent, Facet, Receptacle, EventSource, and others, which are used to specify the use of the CORBA Component platform by an application.

A PM also specifies requirements on the connection and use of the parts of the platform, and the connections of an application to the platform. For example, OMG has specified a model of a portion of the CORBA platform in the UML Profile for CORBA. This profile provides a language to use when specifying CORBA systems.

A PSM in turn may transformed into an even more specific model, tailored to achieve a certain quality of service, perhaps an availability requirement.

The system development process involves transforming a model of a system from one modeling viewpoint to another. The way in which this transformation is done may be guided by standard mappings.

In TOGAF terms, a PSM is also referred to as a *solution building block* model.

Model Transformations

Model transformation is the process of converting one model to another model of the same system.

In order to make the transformation from PIM to PSM, design decisions must be made. These design decisions can be made during the process of developing a design that conforms to engineering requirements on the implementation. This is a useful approach, because these decisions are considered and taken in the context of a specific implementation design. This manual transformation process is not greatly different from how much good software design work has been done for years. The MDA approach adds value in two ways:

8

- the explicit distinction between a platform independent model and the transformed platform specific model,

- the record of the transformation.

A PIM may be prepared using a platform independent UML profile. This model may be transformed into a PSM expressed using a second, platform specific UML profile. The transformation may involve marking the PIM using marks provided with the platform specific profile.

The UML 2 profile extension mechanism may include the specification of operations; then transformation rules may be specified using operations, enabling the specification of a transformation by a UML profile.

It is germane here to point out that MDA helps one build models and use those models in the downstream development process.

MDA Approach

Figure 5 offers a brief summary of the MDA approach to software specification and development. It assumes analysis of the problem space has been performed, that a Computer Independent Model(s) have been developed, and that the effort to develop the Platform Independent Model is ready to be undertaken. It is expected that much of this process will be automated, assisted by MDA tools. At the time of this writing it is difficult to assess whether one or a set of collaborating tools will be required to complete these process; hence, the singular noun, *tool*, will be used. Also, for simplicity's sake only one model at each level of abstraction (i.e., PIM, PSM) is shown. However, it is quite reasonable to expect abstraction within these levels. (For example, a PIM of a generic payment system may be specialized to specific transactional models independent of any transactional product.)

An MDA modeling tool assists the process of transforming the CIMs into UML models capturing the software design of the business processes independent of any commercial or "home-grown" technical solution. The tool will assist the modeler in adding any information (e.g., configuration) need to complete the PIM. The tool also documents the design process, a key to any effective software engineering effort.
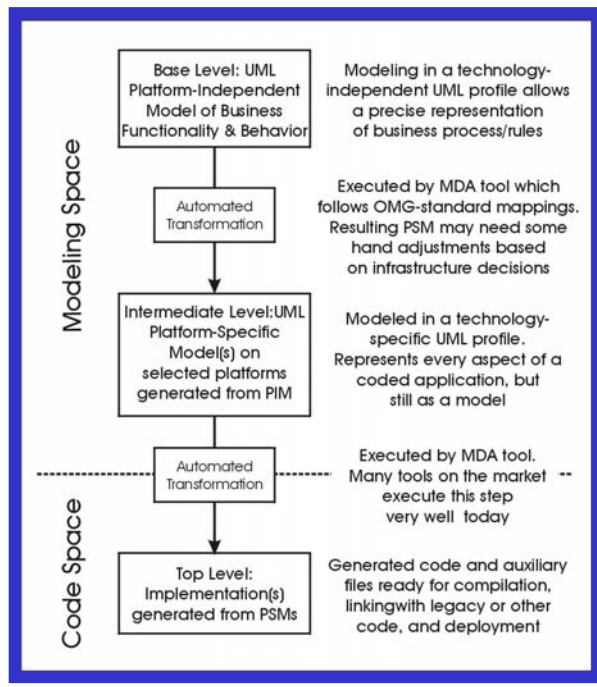
Figure 5: MDA Approach

The next step is the transformation of the PIM to a PSM, using OMG standard profiles that extend UML to embrace the semantics of the target platform. As with the PIM, it is conceivable that levels of abstraction can occur here. For example, a PSM may reflect a generic component-based platform, which in turn could be further specialized to a CORBA Component Model or Enterprise Java Beans.

When completed, the tool enters the code development phase. Again, transformation are performed which generate code and any ancillary files required to produce the technical solution supporting the business processes.

Important !!!

This simple diagram might mislead the reader to the conclusion that this is a waterfall process. Nothing could be farther from the truth. Once verification and validation tests are run on the code, MDA requires any necessary changes be captured in the pertinent models. This ensures that all documentation is synchronized with the final code.

Where the strength lies - TOGAF and MDA working together

"The goal of MDA is to create an enterprise architecture modeling capability..." [4] The TOGAF ADM calls for creating models at phases A, B, C, and D. TOGAF is agnostic about what modeling approach or tool to use and doesn't get into detail on models. It is not difficult to see how certain MDA CIM and PIM models could be used for certain parts of phases A, B, C, and D in TOGAF. Finally Phase G is where the TOGAF ADM does "implementation governance"... where the MDA approach of transformation of CIM models to PIM models and PIM models to PSM models … would improve the integrity of going from architecture to implementation.

So:

Frameworks prescribe the set of deliverables (i.e., all the different *types* of deliverables) that a complete enterprise architecture description should have.

This is something that a given organization will decide, based on its own business environment. In some cases, it may choose a highly generic framework such as the Zachman Framework. In other cases, the vertical sector that the enterprise is in (e.g., Government; Defense; Telecommunications) may have its own generic framework that is mandated or recommended for use by enterprises within that sector.

Neither TOGAF nor MDA is dependent on the choice of framework. So - pick your favorite framework! But to fill that framework out, you'll need a method. The reason for this is that a framework typically does not describe two very important things:

- Which specific deliverables (out of the prescribed complete set) to develop as part of a particular architecture development effort.

- How to develop those specific deliverables – i.e., how to decide what they should contain (frameworks prescribe *types* of deliverables);

Effective ADMs address these problems. They provide guidelines on how to scope a particular architecture development effort; and they provide the means to discover and understand what the deliverables should contain.

The TOGAF ADM does just that: it provides guidelines on scoping the architecture activity; and it provides a step-by-step method for discovering and understanding what the deliverables should contain. TOGAF also suggests the creation of models to capture that understanding in a re-usable form.

However TOGAF does not prescribe the means to use to capture models. What is needed is to pick a modeling approach that is fit for purpose for each of the models that require capturing.

MDA provides the means to capture models, manage those models, translate between models, and deal with downstream generation of code. For many of the models suggested in TOGAF MDA provides a very good fit as depicted in the table below.

Table 1. TOGAF ADM / MDA Synergy Points

	CIM	PIM	PSM	PM	Transformations
ADM Phase A	A	A	NA	NA	NA
ADM Phase B	A	A	NA	NA	NA
ADM Phase C	NA	A	NA	NA	NA
ADM Phase D	NA	NA	A	A	A
ADM Phase E	NA	NA	U	NA	NA
ADM Phase F	NA	NA	NA	NA	NA
ADM Phase G	NA	NA	U	U	U
ADM Phase H	NA	NA	NA	NA	NA

A – Applicable in that TOGAF indicates that models like these be created
NA – Not Applicable only in that these models are not produced at this phase
U – Useful in this phase

Conclusions

The TOGAF Architecture Development Method (ADM) is a detailed, industry standard method for developing an enterprise architecture that addresses the business needs of the organization concerned. It calls for the development of a number of architectural models in order to effectively describe the architecture.

OMG's Model Driven Architecture (MDA) is the industry's leading standards-based approach to model-based system development. It helps ensure that enterprise software is developed in a way that conforms with the architecture, and helps ensure the reusability and easy maintainability of the components of the architecture throughout its life-cycle, thereby completing the cycle from requirements through to implementation and maintenance.

Both of these industry standards are intended to complement the use of frameworks, such as the well-known Zachman Framework, which provide a taxonomy for relating real world business concepts to the concepts of information systems and their implementation.

In publishing this White Paper, the memberships of The Open Group and the OMG are seeking to describe and promulgate the enormous synergy that they believe exists within the industry if these concepts are used effectively together. They also hope to remove much of the confusion that currently permeates the topic of enterprise architecture, and to enable architecture practitioners to see how TOGAF and MDA can be used together with a framework of choice, to bring greater discipline and reusability to this whole field.

The high level relationship between these concepts is simple enough and powerful enough to bear repeating one last time:

Pick your favorite framework, Use TOGAF to fill it up; and use MDA to empty it!

References

[1] Object Management Group. *MDA Guide, Version 1.0,* OMG Document Number: omg/2003-06-01. June 12, 2003.

[2] John Spencer, et al (2004). *TOGAF "Enterprise Edition" Version 8.1* [Online]. Available: http://www.opengroup.org/architecture/togaf8-doc/arch/

[3] J. F. Sowa, J. A. Zachman (1992). *Extending and formalizing the framework for information systems architecture,* IBM System Journal, Vol 31, No 3, 1992

[4] Walter Stahlecker (2003). [Email]. {need to check with Walter}

[5] http://www.opengroup.org/overview/

[6] http://www.opengroup.org/architecture/

Appendix 1 - TOGAF Background

About TOGAF Version 8 "Enterprise Edition"

TOGAF Version 8 "Enterprise Edition" ("TOGAF 8" for short) is a detailed method and set of supporting resources for developing Enterprise Architectures. Developed and endorsed by the membership of The Open Group, TOGAF 8 represents an industry consensus method for Enterprise Architecture that is available for use internally by any organization around the world - members and non-members of The Open Group alike - under a free, perpetual license.

As a comprehensive, open method for Enterprise Architecture, TOGAF 8 complements, and can be used in conjunction with, other frameworks that are more focused on specific deliverables for particular vertical sectors such as Government, Defense, and Finance.

The latest version of TOGAF 8 is Version 8.1, was published in December 2003.

Why do I need enterprise architecture?

The primary reason for developing enterprise architecture is to support the business by providing the fundamental technology and process structure for an IT strategy. This in turn makes IT a responsive asset for a successful modern business strategy.

Today's CEOs know that the effective management and exploitation of information through IT is the key to business success, and the indispensable means to achieve competitive advantage. Enterprise architecture addresses this need, by providing a strategic context for the evolution of the IT system in response to the constantly changing needs of the business environment.

Furthermore, good enterprise architecture enables you to achieve the right balance between IT efficiency and business innovation. It allows individual business units to innovate safely in their pursuit of competitive advantage. At the same time, it assures the needs of the organization for an integrated IT strategy, permitting the closest possible synergy across the extended enterprise.

The technical advantages that result from good enterprise architecture bring important business benefits, which are clearly visible in the bottom line:

- A more efficient IT operation

- Lower software development, support, and maintenance costs

- Increased portability of applications

- Improved interoperability and easier system and network management

- A better ability to address critical enterprise-wide issues like security

- Easier upgrade and exchange of system components

- Better return on existing investment, reduced risk for future investment

- Reduced complexity in IT infrastructure

- Maximum return on investment in existing IT infrastructure

- The flexibility to make, buy, or out-source IT solutions

- Reduced risk overall in new investment, and the costs of IT ownership

- Faster, simpler, and cheaper procurement

Buying decisions are simpler, because the information governing procurement is readily available in a coherent plan.

The procurement process is faster - maximizing procurement speed and flexibility without sacrificing architectural coherence.

Why do I need a "Method" for enterprise architecture?

Using an architecture method will speed up and simplify architecture development, ensure more complete coverage of the designed solution, and make certain that the architecture selected allows for future growth in response to the needs of the business.

Architecture design is a technically complex process, and the design of heterogeneous, multi-vendor architectures is particularly complex. TOGAF plays an important role in helping to "demystify" the architecture development process, enabling IT users to build genuinely open systems-based solutions to their business needs.

Why is this important?

Those IT customers who do not invest in enterprise architecture typically find themselves pushed inexorably to single-supplier solutions in order to ensure an integrated solution. At that point, no matter how ostensibly "open" any single supplier's products may be in terms of adherence to standards, the customer will be unable to realize the potential benefits of truly heterogeneous, multi-vendor open systems.

What is TOGAF?

TOGAF is an architecture method and resource base that enables you to design, evaluate, and build the right enterprise architecture for your organization.

The key to TOGAF is the TOGAF Architecture Development Method (ADM) - a reliable, proven method for developing an IT enterprise architecture that meets the needs of your business.

What kind of "architecture" does TOGAF deal with?

There are four types of architecture that are commonly accepted as subsets of an overall Enterprise Architecture, all of which TOGAF is designed to support:

- A business (or business process) architecture - this defines the business strategy, governance, organisation, and key business processes.

- An applications architecture - this kind of architecture provides a blueprint for the individual application systems to be deployed, their interactions, and their relationships to the core business processes of the organization.

- A data architecture - this describes the structure of an organization's logical and physical data assets and data management resources.

- A technology architecture - this describes the software infrastructure intended to support the deployment of core, mission-critical applications. This type of software is sometimes referred to as "middleware".

Just how do you use TOGAF?

Basically, information about the benefits and constraints of the existing implementation, together with requirements for change, are combined using the methods described in the TOGAF ADM, resulting in a "target architecture" or set of target architectures.

The TOGAF Standards Information Base (SIB) provides a database of open industry standards that can be used to define the particular services and components required in the products purchased to implement the developed architecture. The SIB provides a simple and highly effective way to procure against enterprise architecture.

Appendix 2 - MDA Background

OMG's Generic MDA Standards Process

OMG adopts models that exploit the MDA pattern, to enable portability, interoperability and reusability. Standards are developed through the OMG technology adoption process, or by reference to existing standards.

To be adopted by the OMG, a submitted technology must include a PIM and at least one PSM. In addition, there must be an implementation, or a commitment to provide an implementation within a year, for at least one platform.

Adoptions may also include the PIM-to-PSM mapping used, and the record of the transformation that produced the PSM.

Types of MDA-Related Standard

With MDA, OMG envisages the following distinct types of standard being developed:

Computation Independent Models (also known as Domain Models). There are many existing OMG domain technology adoptions. Examples include:

- Data Acquisition from Industrial Systems (Manufacturing),

- Air Traffic Control (Transportation),

- Gene Expression (Life Science Research),

- Personal Identification Service (Healthcare).

Each has an implicit model, expressed (partly at least) in the Interface Definition Language (IDL) specification of the technology, and use of the interfaces depends on an understanding of that implicit model. Domain technologies adopted in the future are expected to be in the form of normative PIMs expressed using UML, augmented by normative PSMs for at least one target platform.

Platform Independent Models

Several OMG specifications are in various stages of development which will provide generic platform independent models, forming a library of reusable PIMs. Examples include:

- Business Process Runtime Interfaces

- Super Distributed Objects

- Product Lifecycle Management

- Software Defined Radios

Platform Specific Models

Platform specific models may be in the form of UML models, and may be made available in MOF compliant repositories as UML models, MOF models, or models in extended UML or other languages specified using the MOF model (including MOF languages corresponding to UML profiles). As an example, the CWM models provide a rich language for specification of the design and use of data warehouses. Many CORBA-targeted PSMs are also in various stages of development.

Platform Models

In addition to the basic CORBA technology and the CORBA language mappings, OMG has adopted a number of specialized platform technologies. Examples include: Realtime CORBA, Minimum CORBA, Fault-Tolerant CORBA, CORBA Components, and a variety of domain technologies.

Languages

For example, IDL for interface specification; UML for model specification; OCL for constraint specification.

- The Unified Modeling Language (UML) is a standard modeling language for visualizing, specifying, and documenting software systems. Models used with MDA can be expressed using the UML language. A major revision and enhancement to UML, UML2, is in progress at the time of writing. UML2 will integrate a set of concepts for completely specifying the behavior of objects, the UML action semantics.

- The Meta-Object Facility (MOF) technology provides a model repository that can be used to specify and manipulate models, thus encouraging consistency in manipulating models in all phases of the use of MDA.

UML Profiles

Profiles are a UML extension mechanism. A profile in effect specifies a new modeling language, by adding new kinds of language elements and/or restrictions to the original language. The new language may then be used to build a model, or to apply the new or restricted language elements to an existing model.

Several specifications for profiles exist or are soon to be published. They include:

- the UML Profile for CORBA, for use by models specific to the CORBA platform

- the Enterprise Distributed Object Computing (EDOC) Profile, for use in platform independent models for certain classes of component-based platforms.

- The Enterprise Application Integration (EAI) Profile for defining and publishing a metadata interchange standard for information about accessing application interfaces.

Pervasive Services

Pervasive services are those providing computing infrastructure capability over a wide variety of platforms. For example, Naming, Transaction, Security, and Trading. Work is planned to provide these specifications.

Protocols

For example, GIOP/IIOP (both a structure and an exchange protocol), XMI (a structure specification usable as payload on multiple exchange protocols).

Transformation Technologies

- Model Mappings

- Metamodel Mappings

- Model Marking

- Architectural IDEs / MDA Tools

Conformance Testing

To support MDA, the OMG will collaborate with the appropriate organizations and companies to promote conformance testing and certification of products (branding).

Appendix 3 – Glossary and Abbreviations

Acquirer - An organization that procures a system, software product, or software service from a supplier. (The acquirer could be a buyer, customer, owner, user, or purchaser.) (IEEE Std 1471-2000)

API - See Application Program Interface

APP - See Application Portability Profile

Application - A classification of computer programs designed to perform specific tasks, such as word processing, database management, or graphics.

Application Platform - The collection of hardware and software components that provide the services used by support and mission-mission-specific software applications.

Application Program Interface (API) - (1) The interface, or set of functions, between the application software and the application platform.

Application Program Interface (API) - (2) The most common means by which an software programmer invokes other software functions.

Application Software - Software entities that have a specific business purpose.

Architect - The person, team, or organization responsible for systems architecture. (IEEE Std 1471-2000)

Architecting - The activities of defining, documenting, maintaining, improving, and certifying proper implementation of an architecture. (IEEE Std 1471-2000)

Architectural description (AD) - A collection of products to document an architecture. (IEEE Std 1471-2000)

Architectural Framework - A tool for assisting in the production of organization-specific architectures. An architectural framework consists of a technical reference model, a method for architecture development and a list of component standards, specifications, products and their interrelationships that can be used to build up architectures.

Architecture - (1) A formal description of a system, or a detailed plan of the system at component level to guide its implementation.

Architecture - (2) The structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time.

Architecture – (3) The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution. (IEEE Std 1471-2000)

Architecture Continuum - A part of the Enterprise Continuum. The Architecture Continuum provides a repository of architectural elements with increasing detail and specialization. This Continuum begins with foundational definitions like reference models, core strategies, and basic building blocks. From there it spans to industry architectures and all the way to an organization's specific architecture.

Architecture View - A perspective from which architecture may be viewed in order to ensure that a specific topic is considered in a coherent manner - e.g. Security.

Architecture, Baseline - The existing system architecture before entering a cycle of architecture review and redesign.

Baseline - A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development and that can be changed only through formal change control procedures or a type of procedure such as configuration management.

Building Block - A is simply a package of functionality defined to meet business needs. Architectural Building Blocks relate to the Architecture Continuum, and are defined or selected as a result of the application of the Architecture Development Method. Solution Building Blocks relate to the Solutions Continuum, and may be either procured or developed.

Business system - Hardware, software, policy statements, procedures and people that together implement a business function.

Common Object Request Broker Architecture (CORBA) - An OMG distributed computing platform specification that is independent of implementation languages. (MDA Guide)

Common Warehouse Metamodel (CWM) - An OMG specification for data repository integration. (MDA Guide)

CORBA - Common Object Request Broker Architecture.

CORBA Component Model (CCM) - An OMG specification for an implementation language independent distributed component model. (MDA Guide)

ECA - Enterprise Computing Architecture. (MDA Guide)

EDOC - Enterprise Distributed Object Computing. (MDA Guide)

Enterprise - The highest level in an organization ----includes all missions and functions.

Enterprise Continuum - Comprises two complementary concepts: the Architecture Continuum and the Solutions Continuum. Together these are a range of definitions with increasing specificity, from foundational definitions and agreed enterprise strategies all the way to architectures and implementations in specific organizations. Such coexistence of abstraction and concreteness in an enterprise can be a real source of confusion. The Enterprise Continuum also doubles as a powerful tool to turn confusion and resulting conflicts into progress.

Enterprise Java Beans (EJB) - A component standard for the Java platform standardized by the JCP. (MDA Guide)

Enterprise Model - A high level model of an organization's mission, function, and information architecture. The model consists of a function model and a data model.

Function - A useful capability provided by one or more components of a system.

Hardware - (1) Physical equipment, as opposed to programs, procedures, rules, and associated documentation.

Hardware - (2) Contrast with software.

Human Computer Interface - Hardware and software allowing information exchange between the user and the computer.

IDE - Integrated Development Environment. (MDA Guide)

IEEE - Institute of Electrical and Electronic Engineers.

IIOP - Internet Inter ORB Protocol. (MDA Guide)

Information Domain - A set of commonly and unambiguously labeled information objects with a common security policy that defines the protections to be afforded the objects by authorized users and information management systems.

Information System - The computer-based portion of a business system.

Information Technology (IT) - The technology included in hardware and software used for information, regardless of the technology involved, whether computers, communications, micro graphics, or others.

Interface - Interconnection and interrelationships between two devices, two applications, or the user and an application or device.

Interface Definition Language (IDL) - An OMG and ISO standard language for specifying interfaces and associated data structures. (MDA Guide)

Interoperability - (1) The ability of two or more systems or components to exchange and use information.

Interoperability - (2) The ability of systems to provide and receive services from other systems and to use the services so interchanged to enable them to operate effectively together.

Life cycle - The period of time that begins when a system is conceived and ends when the system is no longer available for use.

Life cycle model - A framework containing the processes, activities, and tasks involved in the development, operation, and maintenance of a software product, which spans the life of the system from the definition of its requirements to the termination of its use. (IEEE Std 1471-2000)

Mapping - Specification of a mechanism for transforming the elements of a model conforming to a particular metamodel into elements of another model that conforms to another (possibly the same) metamodel. (MDA Guide)

Meta Object Facility (MOF) - An OMG standard, closely related to UML, that enables metadata management and language definition. (MDA Guide)

Metadata - Data that represents models. For example, a UML model; a CORBA object model expressed in IDL; and a relational database schema expressed using CWM. (MDA Guide)

Metamodel - A model of models. (MDA Guide)

Metaview (also known as a viewpoint) - A specification of the conventions for constructing and using a view. A metaview acts as a pattern or template of the view, from which to develop individual views. A metaview establishes the purposes and audience for a view, the ways in which the view is documented (e.g., for visual modeling), and the ways in which it is used (e.g., for analysis).

Model - A formal specification of the function, structure and/or behavior of an application or system. A model is often presented as a combination of drawings and text. The text may be in a modeling language or in a natural language. (MDA Guide)

Model Driven Architecture (MDA) - An approach to IT system specification that separates the specification of functionality from the specification of the implementation of that functionality on a specific technology platform. (MDA Guide)

Model Transformation - The process of converting one model to another model of the same system.

Object Management Architecture (OMA) - An Object-Oriented Architecture for Distributed Computing that forms the foundation for CORBA. (MDA Guide)

OMA - See Object Management Architecture. (MDA Guide)

Open specifications - Public specifications that are maintained by an open, public consensus process to accommodate new technologies over time and that are consistent with international standards.

Open system - A system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered applications software: (a) to be ported with minimal changes across a wide range of systems, (b) to interoperate with other applications on local and remote systems, and (c) to interact with users in a style that facilitates user portability.

ORB - Object Request Broker

OS - Operating System

OSI - Open Systems Interconnection

Pervasive Service – Software providing computing infrastructure capability over a wide variety of platforms.

Platform - A set of subsystems/technologies that provide a coherent set of functionality through interfaces and specified usage patterns that any subsystem that depends on

the platform can use without concern for the details of how the functionality provided by the platform is implemented. (MDA Guide)

Platform Independent Model (PIM) - A model of a subsystem that contains no information specific to the platform, or the technology that is used to realize it. (MDA Guide)

Platform Model - A set of technical concepts, representing the different kinds of parts that make up a platform and the services provided by that platform.

Platform Specific Model (PSM) - A model of a subsystem that includes information about the specific technology that is used in the realization of it on a specific platform, and hence possibly contains elements that are specific to the platform. (MDA Guide)

Portability - (1) The ease with which a system or component can be transferred from one hardware or software environment to another.

Portability - (2) A quality metric that can be used to measure the relative effort to transport the software for use in another environment or to convert software for use in another operating environment, hardware configuration, or software system environment.

Portability - (3) The ease with which a system, component, data, or user can be transferred from one hardware or software environment to another.

Profile - A set of one or more base standards, and, where applicable, the identification of those classes, subsets, options, and parameters of those base standards, necessary for accomplishing a particular function.

Profiling - Selecting standards for a particular application.

Repository - A system that manages all of the data of an enterprise, including data and process models and other enterprise information. Hence, the data in a repository is much more extensive than that in a Data Dictionary, which generally defines only the data making up a database.

Request for Comment (RFC) -An unsolicited draft specification submitted to OMG TC for standardization. (MDA Guide)

Request for Proposal (RFP) - A document requesting OMG members to submit proposals to the OMG's Technology Committee. Such proposals must be received by a certain deadline and are evaluated by the issuing task force. (MDA Guide)

RM - Reference Model

RMI - Remote Method Invocation - used in Java platforms for remotely invoking methods of a Java Object. (MDA Guide)

RM-ODP - Reference Model for Open Distributed Processing, an ISO set of standards. (MDA Guide)

SOAP - Simple Object Access Protocol - A popular protocol used to remotely invoke operations of a web object across the web. (MDA Guide)

Solutions Continuum - A part of the Enterprise Continuum. The Solutions Continuum contains implementations of the corresponding definitions in the Architecture Continuum. In this way it becomes a repository of re-useable solutions for future implementation efforts.

System - A collection of components organized to accomplish a specific function or set of functions (taken from Draft Recommended Practice for Architectural Description IEEE P1471/D5.2).

System - A collection of components organized to accomplish a specific function or set of functions. (IEEE Std 1471-2000)

System and network management service - A cross-category service of the Application Platform Entity of the Technical Reference Model that provides for the administration of the overall information system. These services include the management of information, processors, networks, configurations, accounting, and performance.

System stakeholder - An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system. (IEEE Std 1471-2000)

Taxonomy of architecture views - Organized collection of all views pertinent to an architecture.

Technical Reference Model - A structure which allows the components of an information system to be described in a consistent manner.

TRM - Technical Reference Model

UML Profile - A standardized set of extensions and constraints that tailors UML to particular use. (MDA Guide)

Unified Modeling Language (UML) - An OMG standard language for specifying the structure and behavior of systems. The standard defines an abstract syntax and a graphical concrete syntax. (MDA Guide)

User - (1) Any person, organization, or functional unit that uses the services of an information processing system.

User - (2) In a conceptual schema language, any person or any thing that may issue or receive commands and messages to or from the information system.

User Interface Service Hardware - A service of the Application Platform entity of the Technical Reference Model that supports direct human-machine interaction by controlling the environment in which users interact with applications.

View – (1) A representation of a whole system from the perspective of a related set of concerns. (IEEE Std 1471-2000)

View – (2) A viewpoint model or view of a system is a representation of that system from the perspective of a chosen viewpoint. (MDA Guide)

Viewpoint – (1) (also known as a metaview) - A specification of the conventions for constructing and using a view. A pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis. (IEEE Std 1471-2000)

Viewpoint – (2) A viewpoint on a system is a technique for abstraction using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within that system. (MDA Guide)

XML Metadata Interchange (XMI) - An OMG standard that facilitates interchange of models via XML documents. (MDA Guide)

Appendix 4 – Comments Received

The following captures edited comments received on the paper to stimulate further discussion and work for the November 18th workshop.

I have skimmed the paper and have concerns about the positioning of MDA as all about developing 'systems': even CIMs are positioned as being about scoping the development of systems. I think that business models have benefit in their own right for managing enterprise architectures - as presumably does TOG. And OMG is developing standards for several of these e.g. Business Rules, Business Process, Organization Structure, Business Motivation (to come).

I think the real synergy is that TOGAF focuses on the method whereas OMG provides standards, supported by technology/tools, for the content/artifacts of what the method produces, and facilitates the transformation of artifacts between the different lifecycle stages, as well as the general management of the artifacts.

It will be good if we highlight the significance of Modeling/MDA for Information/Data Integration as a technique within ADM -that helps align IA with Process/Application Architecture and ITA.

Some of the techniques within OMG's MDA include - CWM/MOF/XMI -- these can be mapped into the Reference Models that ADM proposes - TRM/IIRM and other such repositories.

The Services Taxonomy - categorized as Communications Services, Business Services, Basic/foundation Services, Converged Services, etc. -- can be associated to specific sets of data/object models - such as MOML/MSML to Media Services, SAML to Security Services, ebXML to business Services, CpML/VoiceXML for communication services, DOML for DRM Services, etc., - such mapping will allow for xml based common information model - and their association with Services.

Somewhere the value-proposition of MDA and SOA and the combination for building Convergent Architectures needs to be highlighted http://www.convergentarchitecture.com/ Richard Hubert has applied concepts/principles from both worlds.

Finally a few examples of synergies -like the (seven) ones in my paper/presentation will also help. http://www.opengroup.org/architecture/0404brus/papers/rakesh/mdasoa16.pdf

As an Information Security (INFOSEC) professional, I find the white paper titled "TOGAF ADM and MDA (R) - DRAFT" coming up a little short on the subject of INFOSEC and specifically on the subject of how either the Open Group's TOGAF ADM or OMG's MDA handles the security topic of INFOSEC assurance in an enterprise environment. So my bottom line, will there be more added to the white paper titled "TOGAF ADM and MDA (R) - DRAFT" to address my concerns?

It is my belief that security topic of INFOSEC "assurance" must be addressed by any enterprise development which seeks to migrate from yesterday's technology onward to today's and into the future (whatever the future technology may bring). Therefore, IMHO, the joint white paper needs more information on this topic.

So what do I mean by INFOSEC assurance? How does it apply to the migration of distributed computing in an enterprise environment and the technology migration risks to that enterprise?

I think that whatever the computer industry does to provide freedom from worry is called assurance. Assurance is the set of things the builder and the operator of a system do to convince you that it really is safe to use. Any migration entails risk and must be part of the system development process.

The theory behind assurance is that the computer industry knows what things need to be done to make a system or network secure, can convince you that doing those things will make the

system secure, and can prove that those things have been done for your particular system or network.

If you care about security, someone who wants to sell you a secure system or who wants to convince you to use a supposedly secure system/network has to convince you that:

*   The system/network can enforce the policy you're interested in, and

*   The system/network works!

If you think the system won't support your policy, the seller or operator can simply bring up the system's policy management interface, ask you what your policy is, and try to enter it into the system. If he succeeds, he wins---you're convinced. If not, he loses---no sale.

If you think the system doesn't work, the seller or operator has to convince you that the system keeps its security promises. To do this, he has to make an assurance argument.

Assurance arguments usually try to prove three things:

1.   The system's protection mechanisms are correct (in other words, they're not full of bugs, and they enforce the stated policy).

2.   The system always uses its protection mechanisms when they're needed (for example, it always checks access control whenever a user asks for access to a protected resource).

3.   There's no way to circumvent the system's protection mechanisms (so the system doesn't have any "back doors" which might let people do end-runs around the protection mechanisms).

Winning the assurance argument involves showing that the system has been designed, built, tested, delivered, installed, and configured properly, and that it is being operated properly.

Three kinds of assurance activities contribute to a strong assurance argument:

1.   Design assurance consists of using good security engineering practices to identify important threats and to choose appropriate countermeasures.

2.   Development assurance requires the use of disciplined processes to implement the design correctly and to deliver the implemented system securely and reliably to the customer.

3.   Operational assurance mandates secure installation, configuration, and day-to-day operation of the system.

Assurance activities are necessary but not sufficient for a good assurance argument; there must also be evidence to prove that the activities were performed.

Good records must be kept during every phase of a system's life (design, development, distribution, installation, and operation) so that people who want to evaluate the system can figure out what has been done to make it secure and decide how much faith in the system's security is justified.