A Comparison of System Modelling for Distributed Applications: RM-ODP vs MDA[*]

Twittie Senivongse, Yunyong Teng-amnuay, and Natawut Nupairoj
Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand
Email: twittie.s@chula.ac.th, yunyong.t@chula.ac.th, natawut.n@chula.ac.th

ABSTRACT – Constructing a business application is a big task as issues always rise starting from how to model business requirements to how the system should be deployed. This process involves making decisions on the design, architecture, and several management aspects for the system. As part of our research project to study the design and construction of distributed enterprise applications, we survey trends in computer technology that relate to building of such applications. This paper reports our survey on system modelling for distributed applications. We cover two key standards: the Reference Model of Distributed Processing (RM-ODP) by ITU-T/ISO and the Model Driven Architecture (MDA) by OMG. Under the two standards, we discuss system modelling from various viewpoints as well as modelling languages. The paper concludes with a comparison between these standards and their status of use at present for constructing distributed systems.

KEY WORDS – System modelling, distributed enterprise systems, RM-ODP, MDA

บทคัดย่อ – ในการสร้างแอพพลิเคชันสำหรับธุรกิจในปัจจุบันจำเป็นต้องคำนึงถึงแง่มุมต่างๆ เริ่มตั้งแต่แบบจำลองความต้องการของธุรกิจควรมีหน้าตาอย่างไร ไปจนถึงการใช้งานระบบที่สร้างเสร็จแล้วควรเป็นไปในลักษณะใด ซึ่งกระบวนการนี้เกี่ยวข้องกับการตัดสินใจในด้านการออกแบบ การเลือกสถาปัตยกรรมสำหรับการทำงานของแอพพลิเคชัน และประเด็นด้านการจัดการต่างๆ ภายในระบบ บทความนี้เป็นส่วนหนึ่งของงานวิจัยด้านการออกแบบและสร้างแอพพลิเคชันสำหรับระบบวิสาหกิจแบบกระจาย ซึ่งได้ทำการสำรวจแนวโน้มของเทคโนโลยีคอมพิวเตอร์ต่างๆ ที่เกี่ยวข้องกับการสร้างแอพพลิเคชันดังกล่าว บทความนี้จะรายงานการสำรวจมาตรฐานในการสร้างแบบจำลองของระบบกระจายสองวิธี ได้แก่ อาร์เอ็ม-โอดีพีของไอทียู-ที/ไอโซ และ เอ็มดีเอของโอเอ็มจี โดยจะกล่าวถึงการสร้างแบบจำลองของระบบ และภาษาที่ใช้อธิบายแบบจำลอง รวมทั้งทำการเปรียบเทียบมาตรฐานทั้งสอง และกล่าวถึงสภาพการนำมาตรฐานทั้งสองไปใช้ในการสร้างระบบกระจาย

คำสำคัญ – การสร้างแบบจำลองของระบบ ระบบวิสาหกิจแบบกระจาย อาร์เอ็ม-โอดีพี เอ็มดีเอ

1. Introduction

Enterprise systems today are facing a big challenge in keeping up with rapid advances in information technology. The problem is not how to 'change' to the new technology but rather how to 'evolve' with it. This means enterprises will consider, if possible, how to mix preexisting systems they have invested with the new technology, rather than constructing the whole systems anew.

Our research project works around the issues in the development and evolution of distributed enterprise systems. In fact there are several areas of concern including system modelling, system architecture, security infrastructure, information management, scalability, resiliency, and change management. We identify system modelling and system architectures as the most fundamental building blocks for system construction. The application of these two is driven by business process modelling which identifies various requirements of the enterprise (Figure 1). Constructing an enterprise system is hence a mapping from business process modelling to design and implementation models. The design will be mapped onto target technology at implementation design phase, before the application is coded and tested. Implementation design will be closely related to software architecture of choice and its middleware that govern how the business processes are implemented and deployed.

[*] This research is supported by the Chulalongkorn University-Industry Linkage Research Grant (Fiscal Year 2004)

Other areas, i.e. security, information management, resiliency, scalability, and change management may be considered as basic services of the system that are also driven by business requirements and utilise design and architecture building blocks in their construction.

Business Aspect (Business Process Modelling and Componentisation)	
Modelling Aspect (Methods and Languages)	**Architectural Aspect** (System Architecture Paradigms)

Figure 1. Fundamentals of enterprise construction

This paper presents one part of our survey of the fundamental building blocks. We focus on the modelling aspect of distributed applications and discuss how systems can be modelled and the languages used to describe models. The presentation is as follows. Section 2 describes major standards for distributed system modelling and Section 3 reviews modelling languages. Section 4 compares the standards and discusses their status of use. The conclusion is found in Section 5.

2. Distributed System Modelling

System modelling is part of system development process that describes the templates of the system to be developed [1]. A distributed application has characteristics that differ from those of a centralised counterpart such that functional decomposition of the system is as important as design of other architectural issues such as system and information security, messaging mechanisms, provision for fault tolerance, quality of services, and system management functions. For distributed applications, it is therefore necessary to also integrate these requirements with functional design into the design model.

Major players in distributed system modelling are ITU-T/ISO and OMG. ITU-T and ISO standardised the Reference Model for Open Distributed Processing (RM-ODP) [2] that defines essential elements within a distributed system and a specification framework based on viewpoint modelling. On the other hand, OMG first started as a middleware-centric organisation standardising the Common Object Request Broker Architecture (CORBA) as a truly vendor- and language-independent middleware that allows objects to interoperate smoothly across hardware platform, operating system, and programming language boundaries. OMG is now moving up from application implementation standardisation to application design standardisation with the Model Driven Architecture (MDA) [3] that describes a new model-oriented development process.

2.1 RM-ODP Modelling

Modelling a distributed system in RM-ODP is by describing the system in five different viewpoints [2]. These viewpoints do not represent layers of system architecture but rather the projection of the system from different angles. They in fact should not be thought of as steps in a development process but they are the closest part of RM-ODP to the development process. Five RM-ODP viewpoints are

- Enterprise viewpoint – This is concerned with business activities at high level focusing on purposes, scope, entities and their communities, roles of the entities for the business, and business policies.
- Information viewpoint – This is concerned with semantics of information that needs to be stored and processed in the system. The information is extracted from individual entities and the viewpoint describes information sources, sinks, and flows.
- Computational viewpoint – This is concerned with functional distribution of the system as a set of logical entities or objects which are sources and sinks of information interacting at interfaces.
- Engineering viewpoint – This is concerned with mechanisms and functions required to support distributed interaction including networked infrastructure and abstract machine which carries out computational interaction.
- Technology viewpoint – This is concerned with the choice of technology for the system and how the system is structured in terms of hardware and software components.

The viewpoints are not independent and should be consistent to model a single system. To express viewpoints, modelling languages are required and viewpoint consistency will be checked. Unfortunately, RM-ODP does not define a specific language or notation for expressing these viewpoints. Since viewpoints must be consistent, several attempts have been made to use specification languages to describe viewpoints so that consistency checking can be verified automatically using proof engines [4], [5], [6]. Despite precision and expressiveness, pure formal specifications are not so well-embraced by enterprise designers since they require mathematical knowledge and hence are quite hard to put into practice. Modellers tend to stick with graphical languages for ease of use and the most widely-used today is the standardised Unified Modeling Language (UML) [7]. By using UML, viewpoints can be represented diagrammatically by various UML diagrams with enhancements for distributed system modelling (see Section 3). It is also possible that design tools can automatically or semi-automatically generate formal specifications of the design from graphical models. This will facilitate enterprise designers and at the same time enable formal consistency checking of viewpoints.

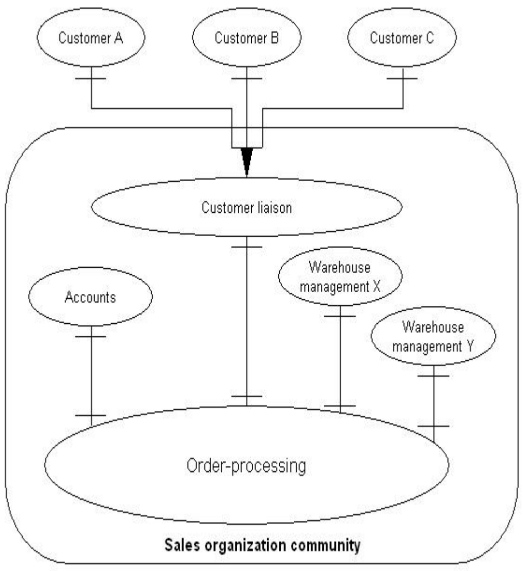

Figure 2. Enterprise viewpoint example [2]

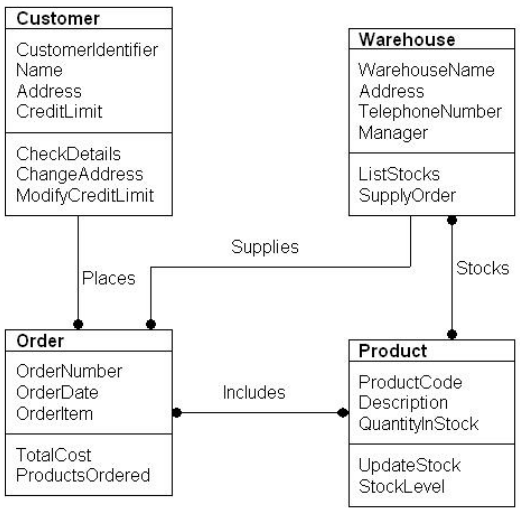

Figure 3. Information viewpoint example [2]

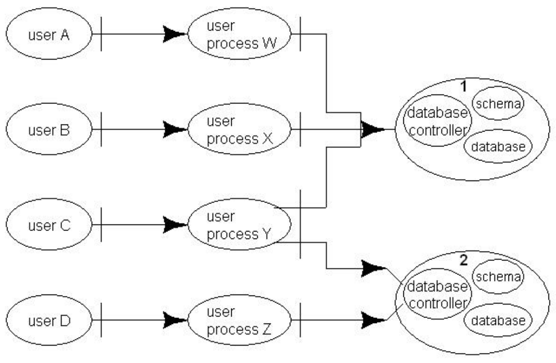

Figure 4. Computational viewpoint example [2]

Figure 2 shows an enterprise viewpoint depicting entities within a sales organisation community. Figure 3 shows an information viewpoint depicting information objects of the sales organisation. A computational viewpoint in Figure 4 shows many users, each corresponding to one of the enterprise entities, and each requiring services that relate to some part of the information schema. The information schema needs to have a shared and persistent representation, so a computational model of database systems interacting with the users via their interfaces is depicted. These examples use a simple diagrammatic modelling notation which is not part of RM-ODP.

RM-ODP also defines functions that are fundamental to the construction of any ODP systems. The functions are base architectural services that will be included in the implementation design. Examples are [8]:

Event notification function – This is concerned with recording of event histories and ordering and notification of events.

Checkpoint and recovery function – This is concerned with checkpointing objects, instantiating checkpoints, and undo or redo interactions for failure recovery.

Replication function – This is concerned with coordination among replica objects and group membership management.

Trading function – This is concerned with advertisement and discovery of interfaces.

Security function – This is concerned with access control, authentication, security audit, key management, and confidentiality and integrity of information.

A number of research [9], [10], [11] argue that object-oriented process, including the well-known Unified Process [1], mostly consider development of centralised sequential systems and rarely suggest how to integrate distributed algorithms to solve problems of distribution such as synchronisation, replication management, or hardware failure. Therefore, development processes have emerged for distributed systems by enhancing RM-ODP which itself lacks concrete development process and design notation. Mostly those research works propose a development process that corresponds to five viewpoints and also proposes their own design notation or merely an extension to UML. In [9], a development process is proposed. Their analysis phase is mapped to the enterprise viewpoint and includes requirement capturing; design phase is mapped to information and computational viewpoints and includes validation of computational objects behaviour and requirements; and finally implementation and testing phases are mapped to engineering and technology viewpoints. The process in [10] defines steps to model structural, instance, interaction, and implementation views of a system. This is analogous to RM-ODP viewpoints as structure and interaction of logical computational objects will be modelled and mapped to components as units of implementation. This process uses an extension to UML, called a UML profile, with specific definitions of stereotypes and tagged values to express particular design. Object Constraint Language (OCL) [12] is also used in this work to express other semantics in the design model such as binding constraint, security policies, and replication policies. For the work in [11], their business modelling corresponds to the enterprise viewpoint; system modelling to information and computational viewpoints; distribution modelling to engineering viewpoint; and implemention to technology viewpoint. Several UML diagrams and OCL are also used to model and formalise their design.

2.2 Model-Driven Architecture (MDA)

Apart from other OMG standards such as CORBA [14] and UML [7], MDA is the next step of OMG to standardise an application development process [3], [13], [15]. As the name implies, everything for an application system is built upon models, i.e. the system is generated from design models. MDA supports the entire life cycle of applications, from design to coding, deployment, maintenance and evolution. It is based heavily on UML and has a close relation to CORBA and other OMG technologies.

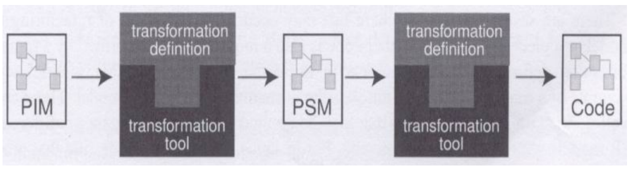

Figure 5. MDA concept [15]

The concept of MDA (Figure 5) is that application systems will be generated from platform-independent design model. The process starts by defining a base Platform-Independent Model (PIM) in UML which expresses only business functionality and behaviour, followed by a second-level PIM which may include some implementation concepts (e.g. transactionality, security level, or configuration information). Then, a standard mapping or transformation definition will be used to map PIM to Platform-Specific Model (PSM). PSM is also a design model but it includes details specific to the underlying implementation platform (e.g. CORBA, EJB, Web Service, .NET) that is chosen for the system, for example, specific classes or interfaces of the platform. PSM will be expressed in dialects of UML called UML profile with addition of stereotypes and tagged values tailored to a particular platform. Application generation is then by mapping from PSM to specific code details of the platform. All of these steps from PIM to PSM and then to code are automatically or semi-automatically done by using MDA transformation tools. Figures 6 and 7 show respectively an example of a PIM for a breakfast delivery service and its PSM for implementing on EJB platform [15].

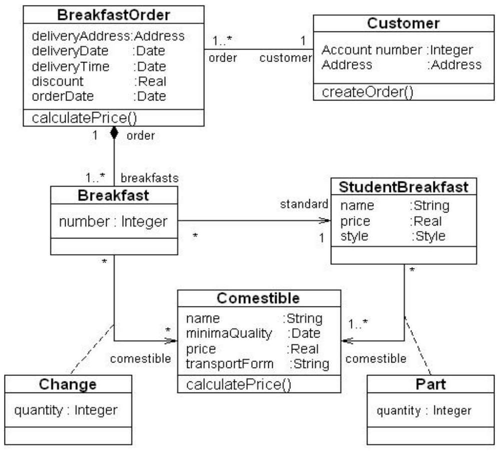

Figure 6. PIM example [15]

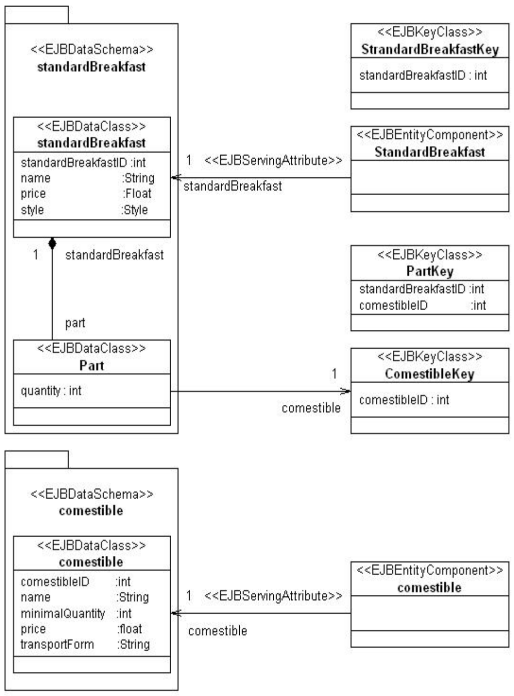

Figure 7. PSM example [15]

With MDA, the application system will evolve gracefully with new technology since the semantics of the application is separate in PIM, and so the system can easily migrate to the new platform by new sets of transformation definitions to PSM and implementation code. In other words, PIM can retain its value over time until it is changed by application requirements. When a part of PIM is changed, only relevant parts of PSM and implementation code will be regenerated. MDA will support integration and interoperability between different technology platforms. Different parts of the application can be mapped onto different platforms and the modelling tools can take care of invocations across multiple platforms.

Other OMG technologies can be complementary to MDA. The design model can incorporate a data repository model represented by the Common Warehouse Metamodel (CWM) [16]. The design can be represented and stored as the Meta-Object Facility (MOF) objects [17] and can be easily ported to other environment in XML Metadata Interchange (XMI) format [18].

MDA will embrace the use of OMG pervasive services (e.g. directory service, transaction service, event service) and vertical domain facilities (e.g. finance, e-commerce) across platform technologies. On building the application, the designer can integrate the model of these pervasive services and domain facilities with base PIM and next-level PIM of the business. By this integration, the designer can make use of the design patterns of these OMG standard services in his/her own design. In fact, the benefits of these services will not be employed if the application platform is not CORBA but OMG is working on transforming the specifications of these services into models so that they can be integrated with PIM and utilised in other platforms also.

Despite its benefits, MDA relies heavily on transformation tools and standard transformation definitions. Whether or not the designer will have to hand-tailor PSM and the generated code depends on the maturity of the standard transformation definitions. MDA assumes that applications are built from scratch and it does not originally aimed for legacy applications. One way to apply it is to wrap legacy applications to the platform technology that will be used first. MDA can generate code to interface with the wrapped parts but the interaction between the wrapped parts and the underlying legacy modules will have to be hand-coded.

The standardisation of MDA is actively in progress as OMG has published an RFP for UML 2.0. The UML profile for CORBA, which is the concept of IDL on UML, was standardised in 2000 while profiles for other platforms are in process. As a result, we can expect MDA with mapping to CORBA middleware to come out in the near future.

2.3 Other Approaches

There are other attempts to propose development processes. The work in [19] proposes an environmental object model in which the system is modelled by objects that are linked into a containment hierarchy by invocation dependency or aggregation. Design constraints, such as synchronisation rules for concurrent object invocations and order of servicing requests, can be put on links. Their process revolves around those environmental objects and takes the waterfall model starting from analysis, logical design, physical design, to implementation. At logical design, environmental classes that provide a solution for functional requirements are defined as classes and interfaces, while at physical design, the logical containment hierarchy is transformed to take into account physical requirements and distributed nature of the implementation platform.

At the other extreme, one can take a formal approach to designing a distributed application as in [20]. Their process starts with writing a requirement specification in TRIO formal language. The specification will be mapped to a high-level design language called TC by identifying data flows, interfaces, attributes and operations semantics, and services and architecture of the underlying implementation platform. From TC, the design can be expressed graphically using TRIO symbols. The advantage of this process is that the design is created from precise and expressive formal requirements.

COMET [21] is another development method for concurrent applications especially distributed and real-time applications. The development process consists of requirement modelling; analysis modelling which emphasises on problem domain classes; design modelling which emphasises on solution domain classes; incremental software construction which includes detailed design, coding, and unit testing; incremental software integration which conducts integration testing; and finally system testing against functional requirements. Its modelling language is UML with an extension to model active objects with their own thread of control as well as synchronous and asynchronous message communication.

3. Modelling Languages

A modelling language is a notation for expressing design and the one that has been widely accepted is the standard UML [7]. UML provides mechanisms to extend the language either by defining stereotypes to specialise existing UML elements for a particular problem, defining tagged values in the form of {property = value} to attribute UML elements, or defining constraints to detail semantics of UML elements. OCL [12] is another way to put formal constraints on the model. It can express guard as invariant and conditions, or parameter-based constraint on the behaviour. Many extensions have been made to UML to define specialised notations for particular characteristics of distributed applications.

3.1 Standard-Based UML Extensions

OMG has issued an RFP for UML profile for Enterprise Distributed Object Computing (EDOC) as there is a requirement for a standard set of enterprise modelling notation, and the submission so far has embodied RM-ODP modelling approach within EDOC [22]. Meanwhile, several research works have put efforts on RM-ODP enterprise modelling based on UML diagrams.

In general, aspects of a distributed system can be modelled using various UML diagrams with special semantics added on by UML extension mechanisms and OCL. Based on RM-ODP concept, the work in [9] collectively expresses viewpoints using standard UML diagrams and OCL to formalise semantics as follows:

- Use case diagrams can be used to capture business requirements.
- Class diagrams can be used to capture information and business objects.
- Activity diagrams can be used to capture business processes.
- Statechart diagrams can be used to capture dynamic nature of information objects.
- Collaboration diagrams can be used to capture configuration and distribution of computational objects.
- Sequence diagrams can be used to capture interactions between computational objects.
- Package diagrams can be used to capture logical architecture and structure in the system.
- Deployment diagrams can be used to capture configuration of system hardware and software components.

A number of research works have specifically attempted to express ODP enterprise viewpoint with UML. In [23], an enterprise entity is represented by a UML object, its action is specified by a UML operation, and its role is represented by an object class with a stereotype <<role>>. Constraints on operations and policies are represented by notes. A community within the enterprise is modelled by a collaboration diagram. An objective of a community is represented by a use case and the enterprise is by a use case diagram. As in [24], details of the enterprise viewpoint can be added by a sequence diagram representing interaction between roles of entities, and an activity diagram (with swim lanes) representing a group of concurrent actions among roles of entities. The work in [25] discusses problems with UML when used to specify enterprise policies. For example, pre/post-conditions in UML which specify policies that constrain actions do not yet support exception-based view of constraints and courses of action when violations occur. Also, the idea of enterprise roles is actually closest to the concept of UML actors, but the UML definition leaves actors outside the domain being modelled, whereas roles should be part of the modelled enterprise. Even so, modellers find UML useful and try to get round these drawbacks while hoping that the raised issues would merit further discussion among UML experts.

For a distributed system with real-time requirements, the standard UML profile called Real-Time UML can be used to model the system [26].

3.2 UML for Distributed Characteristics

Research works have undergone on how to detail the design of a distributed system with semantics relating to distributed characteristics, e.g. security, fault tolerance features, replication, and QoS. Mainly, the design specification is based on the extension to UML diagrams with formal behavioural specifications.

Object Security Constraint Language (OSCL) is proposed in [27] for security specification for information within the system. By modification of some OCL definitions, OSCL can be used to specify security levels (e.g. Unclassified, Confidential, Secret, TopSecret) for classes, attributes, methods, and associations. Objects usually inherit security level of their classes, and rules governing relations for classes and associations can be defined. For example, the security level of a class can be determined by that of some attributes of the class, the subclass must have more restrictive security level than the superclass, or the association must have higher security level than the classes to which it is related.

Specification of a fault tolerant system is described in [28]. This work is based on fault tolerance function in RM-ODP which includes policies for checkpoint/recovery, replication, and event notification. A set of computational objects that enables this function can be represented by RM-ODP based UML (e.g. EDOC), and constraints representing policies are specified by attribute values or OCL expressions for attributes of policy objects. An example of checkpoint/recovery is given where its policy concerns behaviour such as when, where, and what to checkpoint or recover. In [29], the work introduces a replication language called JReplica. Based on Aspect-Oriented Programming paradigm for Java, JReplica enables separation of replication code from functional code of objects within the system. A JReplica aspect, associated with an object class, specifies attributes, state, guard, and actions for the replication of objects of the class. The attributes define the replication policy while the state represents object state to be replicated. The operations define methods to manipulate replicated state with the guard that must be true before executing replication. The actions define what must be done before or after replication. UML is also extended to model replication aspects such that an aspect associated with a particular object class is modelled by a class with a stereotype

<<Replication>>. The Replication state is introduced to the statechart diagram where a guard corresponds to state transition and before/after replication actions are represented in entry/exit actions of the Replication state.

QoS is performance-related requirements that very much varies between domains but is being integrated into frameworks such as RM-ODP and CORBA [30], [31]. UML diagrams can be used to depict QoS specification at different levels such as those discussed in [32]. This work presents how to model QoS requirements and QoS support for the system with UML according to each RM-ODP viewpoint. In the enterprise viewpoint, QoS-related constraints or notes can be attached to the relationships between actors and use cases, and between actors and high-level objects in a collaboration diagram. In the information viewpoint, QoS constraints can be defined on guards, activities, and entry/exit actions on state transition of information objects. For the computational viewpoint, a sequence diagram and a statechart diagram support specification of time constraints on interaction between objects and on state transition respectively. Since UML does not support stream (i.e. continuous media), systems that require to model stream may add a boolean attribute isContinuous to messages. Finally in the engineering and technology viewpoints, QoS negotiations can be specified as collaboration diagrams of QoS patterns which can be applied to engineering components and mapped to a component diagram for implementation technology. In [33], a framework to manage a real-time system to reduce timing faults is considered. In this framework, temporal QoS is defined as tagged values, stemming from Real-Time UML, where the value could be a list of values and include operators. Examples of QoS attributes are QoS for methods which includes worst case execution time and shortest waiting time. QoS for messages includes message deadline, message importance which influences the scheduling of message execution, accepted degradation, e.g. accepted number of execution per number of request messages, and so on.

4. RM-ODP vs MDA

In previous sections, we mainly have discussed RM-ODP and MDA approaches to system modelling and the use of languages such as UML to express models. As already mentioned, other proposals for development processes and formal or graphical modelling languages are around, but they remain as research efforts and have not been put into real use. In this section, we continue with RM-ODP and MDA and compare them on the following aspects:

- Is a development process

RM-ODP No, it is not by definition. It needs a process that will create modelling artifacts in five viewpoints.

MDA Yes, it describes steps to build an architecture, but it emphasizes on design and coding phases. Other phases in the development are implicit.

- Define different views of the system

RM-ODP Yes, the system can be viewed from high level to low level in five angles.

MDA Yes, the system can be viewed from high level to low level, and also in each level, it can be modelled in different angles (e.g. we may model PIM with class diagrams for system structure and sequence diagrams for interaction flows).

- Come with a modelling language

RM-ODP No, it needs other modelling languages such as UML or formal specification languages to model its five viewpoints.

MDA Yes, the primary modelling language is UML but other languages are applicable as well.

- Support modelling of distributed system characteristics

RM-ODP Yes, by using UML profiles or OCL.

MDA Yes, by using UML profiles or OCL.

- Support reuse of patterns in modelling

RM-ODP Yes, models of basic ODP functions are available and can be integrated with the system model.

MDA Yes, models of CORBA pervasive services and vertical domain facilities are available and can be integrated with the system model.

- Complexity

RM-ODP Consistency between viewpoints and how different viewpoints can represent a single system are the main complexities.

MDA Analogously to RM-ODP, consistency between models and between UML diagrams that represent a single level of the model are the main complexities.

- Automate development

RM-ODP No.

MDA Yes, it supports automatic or semi-automatic generation of models and code.

- Application

RM-ODP Aims at large-scaled applications for any application domains.

MDA Also aims at large-scaled applications for any application domains.

- Accepted by distributed system community

RM-ODP Yes, but mainly research community especially in Europe.

MDA Yes, in research and industrial community worldwide.

- Strong points

RM-ODP Research is around for a long while and has influences on standard technologies that support MDA (e.g. CORBA, UML profile for EDOC).

MDA Flexible system modelling and evolution is emphasized. It is embraced by industry and is built from other industry standards. A lot of supporting tools will come out.

- Weak points

RM-ODP It addresses only system modelling, not system evolution. It has no modelling language so it is hard to adopt by industry.

MDA Still immature in research and practice. No tools to support full development and no real example that realises the whole process yet.

It is difficult to say which is better between RM-ODP and MDA, but that is not important. With the current focus of research and industry community on MDA, it may be advantageous if system designers beware of this new promising idea and start exploring and adopting it. Even though system designers will choose MDA over RM-ODP, the latter will still be around. As mentioned earlier, RM-ODP influences several technologies that support MDA. The task forces that standardised those technologies even had previously developed and researched in RM-ODP. Recently, RM-ODP viewpoints have been mapped to PIM and PSM of MDA [34]. That is, enterprise, information, and computational viewpoints can be mapped to PIM level, and engineering and technology viewpoints to PSM level. This means RM-ODP modelling can be adopted by MDA. The two modelling approaches, therefore, will complement each other rather than compete with each other.

5. Conclusion

This paper has discussed about RM-ODP and MDA as key system modelling for building enterprise distributed systems as well as UML as the major modelling language. The survey has pointed out that modelling enterprise distributed systems is a very much active research area as substantial efforts have been made to identify drawbacks of general development processes and a number of specialisation have been proposed to accommodate specific characteristics of distributed systems. Standard bodies are actively standardising processes and modelling languages for distributed applications and the final outcome will lead to the adoption of the standard into commercial CASE tools. However, standard modelling and tools are just 'helping hands' as they only provide guidelines and facilities for designing and constructing the system. Good development still requires experiences to project precisely the requirements and the solution that answers them.

References

[1] I. Jacobson, G. Booch, and J. Rumbaugh, "The Unified Software Development Process", Addison Wesley, 2000.

[2] ITU-T/ISO, "ITU-T X.901| ISO/IEC 10746-1 ODP Reference Model Part 1: Overview", 1995.

[3] J. Siegel et al., "Developing in OMG's Model-Driven Architecture", OMG White Paper, November 2001.

[4] M.W.A. Steen and J. Derrick, "Formalising ODP Enterprise Policies", Proceedings of the 3rd International Enterprise Distributed Object Computing Conference, Mannheim, Germany, September 1999, pp. 84-93.

[5] E. A. Boiten et al., "Constructive Consistency Checking for Partial Specification in Z", Science of Computer Programming, 1999.

[6] H. Bowman et al., "Viewpoint Consistency in ODP: A General Interpretation", Proceedings of the 1st IFIP International Workshop on Formal Methods for Open Object-Based Distributed Systems, Chapman & Hall, March 1996, pp. 189-204.

[7] G. Booch, J. Rumbaugh, and I. Jacobson, "The Unified Modeling Language User Guide", Addison Wesley, 1999.

[8] ITU-T/ISO, "ITU-T X.903| ISO/IEC 10746-3 ODP Reference Model Part 3: Architecture", 1995.

[9] M. Born and A. Hoffmann, "An Object-Oriented Design Methodology for Distributed Services", Proceedings of the 28th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS 28), 1998, pp. 52-64.

[10] M. Born, E. Holz, and O. Kath, "A Method for the Design and Development of Distributed Applications Using UML", Proceedings of the 37th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS-Pacific 2000), Sydney, Australia, November 2000, pp. 253-264.

[11] J. Oldevik and A.-J. Berre, "UML-Based Methodology for Distributed Systems", Proceedings of the 2nd International Enterprise Distributed Object Computing Conference (EDOC'98), La Jolla, CA, USA, November 1998, pp. 2-13.

[12] J. Warmer and A. Kleppe, "The Object Constraint Language Second Edition: Getting Your Models Ready for MDA", Addison Wesley, 2003.

[13] D.S. Frankel, "Model Driven Architecture: Applying MDA to Enterprise Computing", Wiley, 2003.

[14] F. Bolton, "Pure CORBA", Sams, 2001.

[15] A. Kleppe et al., "MDA Explained", Addison Wesley, 2003.

[16] OMG, "Common Warehouse Metamodel Specification V1.1", Document formal/2003-03-02, 2003.

[17] OMG, "MOF Version 1.4", Document formal/2002-04-03, 2002.

[18] OMG, "XML Metadata Interchange (XMI) V2.0", Document 2003-05-02, 2003.

[19] D.I. Donaldson and J.N. Magee, "Distributed System Design Using CORBA Components", Proceedings of the 30th Hawaii International Conference on System Sciences Vol. 1, 1997, pp. 4-13.

[20] M. Pradella et al., "A Formal Approach for Designing CORBA Based Applications", Proceedings of the 2000 International Conference on Software Engineering, 2000, pp. 188-197.

[21] H. Gomaa, "Designing Concurrent, Distributed, and Real-Time Applications with UML", Addison Wesley, 2000.

[22] OMG, "A UML Profile for Enterprise Distributed Object Computing, Joint Final Submission Part II", Document ad/2001-08-20, 2001.

[23] X. Blanc, M.P. Gervais, and R. Le-Delliou, "Using the UML Language to Express the ODP Enterprise Concepts", Proceedings of the 3rd International Enterprise Distributed Object Computing Conference, Mannheim, Germany, September 1999, pp. 50-59.

[24] J.O. Aagedal and Z. Milosevic, "ODP Enterprise Language: A UML Perspective", Proceedings of the 3rd International Enterprise Distributed Object Computing Conference, Mannheim, Germany, September 1999, pp. 60-71.

[25] P.F. Linington, "Options for Expressing ODP Enterprise Communities and Their Policies by Using UML", Proceedings of the 3rd International Enterprise Distributed Object Computing Conference, Mannheim, Germany, September 1999, pp. 72-82.

[26] B.P Douglass, "Real-Time UML, 2nd Ed.", Addison Wesley, 2000.

[27] W. Fernandez-Medina, M. Piattini, and M.A. Serrano, "Specification of Security Constraint in UML", Proceedings of the 35th IEEE International Carnahan Conference on Security Technology, 2001, pp. 163-171.

[28] J. Putman, "Model for Fault Tolerance and Policy from RM-ODP Expressed in UML/OCL", Proceedings of the 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing 2000 (ISORC 2000), Newport, CA, USA, March 2000, pp. 189-196.

[29] J.L. Herrero, F. Sanchez, and M. Toro, "Fault Tolerance as an Aspect Using JReplica", Proceedings of the 8th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS 2001), Bologna, Italy, October 2001, pp. 201-207.

[30] ISO/IEC, "ISO 6-10 Working Document on QoS in ODP", 1997.

[31] C. Sluman et al., "Quality of Service (QoS)", OMG Green Paper om/97-06-04, 1997.

[32] J.O. Aagedal and A-J. Berre, "ODP-Based QoS-Support in UML", Proceedings of the 1st International Enterprise Distributed Object Computing Conference (EDOC'97), Gold Coast, Australia, October 1997, pp. 310-321.

[33] J.L. Contreras and J.L. Sourrouille, "A Framework for QoS Management", Proceedings of the 39th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS 39), Santa Barbara, CA, USA, July 2001, pp. 183-193.

[34] R. Bendraou et al., "From MDA Platform-Specific Model to Code Generation: Coupling of RM-ODP and UML Action Semantics Standard", Proceedings of the 2004 International Conference on Software Engineering Research and Practice (SERP'04), Las Vegas, Nevada, USA, June 2004.

Twittie Senivongse received her B.Sc. in Statistics (2nd class honour) from Chulalongkorn University in 1989, M.Sc in Computing Science from University of London (Imperial College), UK in 1992 and Ph.D. in Computer Science from University of Kent, UK in 1997. She is at present an associate professor at Department of Computer Engineering, Chulalongkorn University. Her main research interest is distributed object technology including middleware architectures and management issues such as service discovery, service evolution, and information systems integration.

Yunyong Teng-amnuay works on varied topics in information systems: operating systems, distributed systems, security, library automation, and networking. He is the coordinator of the Information Systems Engineering Laboratory (ISEL) of the Department of Computer Engineering, Chulalongkorn University. He graduated from Master of Computer Science from Chulalongkorn University in 1979 and Ph.D. in Computer Science in 1984 from Iowa State University.

Natawut Nupairoj received B.Eng. Degree in Computer Engineering from Chulalongkorn University, Thailand, in 1990. He received the M.S. and Ph.D. degrees in Computer Science from Michigan State University, USA, in 1993 and 1998 respectively. Currently, he is a lecturer at Department of Computer Engineering, Chulalongkorn University. His research interests include Parallel Processing, Distributed System, and Computer Networks.