UML Profile for Message-oriented Middleware

Štěpán Vacek
University of Hradec Kralove
Faculty of Informatics and Management
stepan.vacek@gmail.com

Abstract: This paper is focused on approaches to design integration applications based on MOM (Message-oriented middleware). At the beginning it is described a basic overview of modeling and explained the meaning of UML (Unified Modeling Language) profiles. Then it is present fundamental visions of MOM profile including a short description of important parameters which are relevant for messaging platforms. The theoretical introduction is followed by the simple application of the profile in practice. Although the profile is still in progress, it is used for analysis and design of integration applications. Its application provides many benefits such as a centralization of all important properties of a model in a one repository.

Key words: Modeling, Unified Modeling Language, Profile, Message-oriented Middleware, Messaging, Enterprise Application Integration, Enterprise Architecture

1. Introduction

An approach to analysis and design of applications, situated to an integration tier with a messaging, has a lot of similar features like other types of applications which usually use a common two or three-tier architecture. During projects we can exploit familiar methodologies such as RUP (Rational Unified Process), TOGAF (The Open Group Architecture Framework) or agile methodology (e.g. SCRUM). The modeling of systems is typically realized with UML and BPMN (Business Process Modeling Notation).

However, this approach isn’t completely sufficient because it doesn’t reflect a different form of an integration tier. First of all it is necessary to notify that in practice the integration of applications are always tightly coupled with a communication platform. It is not considerable if applications use TPM (Transaction Processing Monitor) or MOM. Basically, UML is a platform independent modeling language, but exactly specifics features for a particular type of product or a platform are very important – in special cases they are even crucial.

2. UML profile

A profile is a common extension mechanism for UML [1] and provides a possibility to modify models for a specific domain or a technology platform. Usually it contains a collection of stereotypes and custom values (tagged values) which creates a logically unit – called a profile. The most common used profiles are created by OMG (Object Management Group) consortium, sometimes in cooperation with other subjects (e.g. companies, universities):
- **UML Profile for CORBA (Common Object Request Broker Architecture)** – It defines a semantic for CORBA IDL (Interface description language). The final code is typically generated from a model by a particular implementation which uses this profile. CORBA is the standard describes communication between heterogeneous distributed applications.

- **UML Profile for EAI (Enterprise Application Integration)** – This profile describes the domain of enterprise application integration (including queues/topics, messages and interactions between participants).

- **UML Profile for EDOC (Enterprise Distributed Object Computing)** – This profile formulates rules for modeling of distributed components. Principally, it is a simple envelope for other specifications, such as ECA (Enterprise Collaboration Architecture), FCM (Flow Composition Model), etc.

It is straightly offered to use the EAI profile for modeling integration applications. Unfortunately, the last specification of the profile [2] is more than six years old. Also it exists only in the phase of published specification without any formal implementation of a meta-model. The specification contains only a common meta-model of EAI and an XMI (XML Metadata Interchange) source is available only for members of the OMG working group. Any proprietary implementation of the EAI profile still doesn’t exist. Therefore this profile, at least from my point of view, doesn’t have a clear future.

The next disadvantage is absolute absence of specific elements which are bounded with communication platforms such as JMS (Java Message Service) or MQ (WebSphere MQ). For example major messaging systems work with expiration and persistence of messages or customized extensions of header. These fields are missing in the specification. Nowadays the specification can understand as a recommendation for vendors of integration solutions or companies which develop their own integration profiles.

### 3. MOM profile

#### 3.1 Motivation

These days I research in the area of designing an integration tier which used MOM as a communication channel. I would like to extend UML with a custom profile which covers modeling of integration flow (mediation) in design and analysis phases. This profile will provide a facility to catch properties and relations between entities related to MOM domain. Also it will be possible to simplify a revision of changes into design documentations emerged during a development.

I have begun to create my own profile for modeling mediation because contemporary UML diagrams (especially activity diagram) don’t provide sufficient facilities to express specifics properties. However, these properties are definitely crucial and they have significant influence on a final form of application architecture. The next reason is the requirement to have a central repository which stores all details and features of models in a one place. This is necessary to automate creation of documentation and generation of interface or code, e.g. WSDL (Web Services Description Language) or XSD (XML Schema).

UML Profile for MOM is designed for the enterprise organization with an information infrastructure which contains some tens of large systems and databases. The
environment is heterogeneous with different software and hardware platforms (including AS/400 or SPARC), protocols (most often XML/MQ and SOAP/HTTP) or data structures (e.g. XML, SWIFT, COPYBOOK). WebSphere MQ has been chosen as a back-bone communication network, followed MOM principles, and WebSphere Message Broker is used as an ESB (Enterprise Service Bus) – both products are delivered by IBM. The company doesn’t have any BPM (Business process management) platform yet, however it is not possible to exclude a process engine into a future.

Modeling is realized mainly in CASE (Computer-aided software engineering) tool Enterprise Architect. A design of the integration tier is stored partly as a document file in Microsoft Word and a model of mediations is created in Microsoft Visio. Specific parameters, related to communication or mediation, are filled directly into documentation. As a result we have an inconsistent model which is stored in separated places: a repository of CASE tool and documents (Microsoft Word/Visio). There are some disadvantages because the main repository is not complete and models of mediations or specific parameters, associated to services and communication, are missing.

The main goal of my work is to create a profile, which full-fills requirements of the particularly organization. Phases of analysis and design will be completely realized from beginning to end in UML and the repository should contain all requested data (model of front-ends, back-ends and a middle-tier). The integration tier will be designed in UML with the applied profile for system integration. The profile should contain major meaningful EAI/MOM elements and in cooperation with other CASE extension should provide a possibility to generate documentation and interfaces (e.g. configuration scripts, XSD, WSDL).

### 3.2 Methodology

It is not easy to start with the creation of a UML profile. As assistance to create a common information system we could use some development methodologies. However, in this case it is not possible, because as far I know any doesn’t exist. Therefore I have set up my custom approach (see Fig.: 1) based on these steps: domain definition, creation, application and evaluation. After the last step, result evaluation, I could propose some revisions based on project experiences or colleague’s feedback.

![Methodology to create a UML Profile](image)
During the activity *domain definition* it is necessary to gather detail information about a particularly domain (or an area in closer meaning). As inputs could be used standards, specifications, experiences and discussions with domain experts. All information has to be evaluated, because we are interesting only in related information which should be a part of the profile. The inputs for my profile were the messaging specification (especially JMS [3], MQ [4]) and my previous project’s experiences, get during several integration large-scale projects. The output contains a list of objects (and their properties) which will be used in the profile.

The following step *creation* uses as an input the result from the previous activity. The list of objects is transformed into a conceptual model. We apply meta-modeling techniques – the objects are extend by particularly meta-classes and properties are filled as theirs meta-attributes. This forms the final UML profile. The result of my profile is displayed bellow (see Fig. 2).

The next step *application* can be understand as the applying of a profile in practice. The result is a model which uses the developed profile. In my case it is the model represented in the image shows the acticity diagram with the applied profile (see Fig. 3).

The final step is the *evaluation* which determines, based on the previous steps (profile application), a quality of the profile. After this step we should know if the profile has enough quality to real use or it has to be updated base on previous experiances.

### 3.3 Conceptual model

The conceptual model, shown below (see Fig. 2), illustrates the basic important entities for models describe integration applications based on MOM. The structure of the conceptual model is complied with the standard Meta-object Facility (MOF) described in detail in the specification [5]. The present version of the profile doesn’t contain all elements which are used in MOM implementations. It is preferred to use only the most important features and keep the profile easy to use as much as possible.

*Mediation* and *Connector* (descendants of *Component*) perform as participants of communication that provide an integration and application logic. *Topic* and *Queue* (descendants of *Channel*) represent a communication channel for distribution of messages. Finally, *Message Flow* connects all objects of model together and creates relation between them.
3.3.1 Component

The abstract object represents a service which is provided either as a mediation flow or as a connector. Also it is possible that a new specific descendant will be created in the next future.

Descendants of Component may have set the following properties:

- **communication** (synchronous, asynchronous or not applicable)
  MOM is asynchronous in nature. Of course, it can be used for synchronous communication when a response is associated with a request according to a correlation identifier (CorrelationId).

- **processing** (transaction, batch)
  Processing can be divided according to mode. Transaction is used for on-line processing, usually in synchronous communication. Batch is used for off-line (mainly asynchronous) processing, in general it is used for bulk processing or synchronization.

- **security**
  Services can be secured by a different approach. Very often a security is not applied at all, because all relevant parts are members of a one infrastructure which is secured on a network level as a demilitarized zone. Otherwise, restrictions can be applied to a specific access point in a network or it is required an authentication (username/password or certificate).

- **load balancing** (yes, no, not applicable)
  A flag whether the processing is distributed to more servers.
3.3.2 Mediation

Mediation contains integration logic such as convergence, transformation or composition of individual messages. Processing is usually described in detail in other diagrams (e.g. activity diagram, sequence diagram), because from the perspective of MOM parameters we are interesting only in routing, publishing/subscribing queues and interactions with adapters or next mediations.

- response time
  The element represents an interval of mediation (in seconds).
- volume
  The volume is a throughput of messages that the system is able to process.
- filtering
  The component can realize filtering by content, either based on specific parameter values or more complex rules.
- monitoring (common, custom, not applicable)
  From the perspective of maintenance, it is important whether a mediation flow sends monitoring messages. The integration layer usually defines common rules for monitoring or a component can use its own approach (custom monitoring). Also this parameter can be not applicable. This situation is applied when monitoring of integration tier is ensured by ESB.

3.3.3 Connector

Connector is a gateway between MOM and back-end systems. It can be supplied as a part of an ESB or developed independently (whether by an external vendor or self-development).

3.3.4 Channel

Channel performs a temporary storage for messages. Its properties determine where it is positioned (name, queue manager) and how it works (persistence).

- name
  A specific queue name (convention should be compliance with a particular platform).
- queue manager
  Queue manager is applied only for some platforms – especially WebSphere MQ [4]. This feature is used when a messaging server is distributed over more physical servers.
- persistence (persistence, non-persistence)
  Persistence has an impact on performance and disaster recovery. This property can be set on two levels, either Channel or Message Flow (described below). Generally, non-persistence processing is much faster because a server keeps messages only in memory and not writes them into storage (a disk or a database). On the other hand if a server fails then un-processing messages will be lost. Persistent processing is used if the reliability of messages is required and lower performance of a server is acceptable. Non-persistent processing is suitable for messages that expire and are not critical in terms of business (logging, monitoring, heart-beat, etc.). Also non-persistent messages can be used for synchronous communication –
it is necessary to process a request in an interval, when the interval is reached then messages should be discarded.

3.3.5 Topic

Topic is a channel for publish-subscribe communication pattern. This approach works as a broadcast when a published message is sent to all subscribed consumers.

- durability
  This parameter affects message delivery, if a message is sent only to active subscribers or to all subscribers (including subscribers which are not temporarily connected, however these have to be subscribed before publishing a message). By default a message is sent only to active consumers. Durability is differed from persistence – durability is a type of delivery to subscriber and persistence is a type of storage on a server.

3.3.6 Queue

A queue performs as a channel for point-to-point communication pattern. In this case multiply producers can publish messages and multiply consumers can consume messages. Each message can be processed by only one consumer.

- type (request, response, not specified)
  Type of a queue should conform to a purpose of communication. Requests and responses should be forwarded through various queues. However, the profile respects the approach of mixed queues as well. This property is focused only on a logical view – technically, there isn’t a difference between a request or response queue.

3.3.7 Message Flow

Message Flow represents a relation between particular components (mediation, connector) and channels (topic, queue). The parameter corresponds to properties set up on a message level:

- message type
  Message type is basic element for a content-based routing. Dispatching is realized according to a value in a message header and parsing of a message body is not required.

- header version, message version
  Mediations can work with multiply version of messages. Each version can be routed or processed in a different way.

- encoding
  Encoding of a message content – most commonly used UTF-8 (in special cases can be differed, e.g. mainframe applications usually work with Latin characters).

- persistence (persistence, non-persistence)
  This parameter can be set up on two levels as was already mentioned above. Each implementation works with this property differently, therefore in a design phase it is necessary to check a product documentation. We must be careful in overwriting of properties. What happens if a channel and a message have different values? This should be answered by product documentation.
- expiry (application defined, defined specified queue, queue defined unlimited)
  Duration of messages is set along with persistence because similar rules are applied. If messages have to be processed (response time is not important at this moment) then expiration is not set. Expiration is usually used for synchronous communication or messages where relevance is limited. Defining a duration can prevent overfilling a queue (e.g. application sends synchronous requests, but for some reason answers are not processed).

4. MOM profile in practice

Currently, I have prepared a basic vision of how the profile should look like. Also I have created an initial version of the profile. To verify a quality of the UML profile, I decided to try it for modeling of mediations on the real small-scale project. This project contains overall 6 mediations:

- 3 simple – The flows perform as a gateway and transform communication between front-end and back-end systems.
- 2 composite – These flows ensure compositions of back-end systems and aggregate results.
- 1 complex – The only one flow provides complex processing, contains more interactions with back-end systems and manages processing according to a received result.

Design output will be used to check a quality and will be helpful to improve the profile. It is necessary to notice that the sample below isn’t a complete design of mediations and describes only the course of my research.

4.1 Mediation model

The profile also contains several entities that are characteristic for integration applications based on MOM. The activity diagrams (with applied MOM profile) contains a routing model of messages between queues, specifics parameters, complex compositions and interactions with back-end systems, connectors or other mediations. Also the profile allows catching detail information about translation of protocols or data structures.

The sample describes the model of composite service. The mediation works with two queues: one for requests and one for responses. When a producer sends a request with a predefined response queue then a response is sent to this queue. (Therefore, the queue for responses has assigned the name REPLY_QUEUE in the model.)

The mediation has three interactions with two connectors, ALPHA and BETA. These connectors represent façades for back-end systems and specific meaning is not important. Crucial is the description of communication through various message types and queues. The orders of processing and conditional branching are described in other diagrams (especially in the common sequence diagram).
Fig. 3 Activity diagram with the applied profile (author's research)

4.2 Central repository

By one of reasons it was to unify all models into a one place that is the central repository of CASE tool Enterprise Architect. This was already achieved in the presented version of the profile. It is illustrated in the image below (see Fig. 4) which contains the toolbox panel for MOM profile and the objects of the model in the repository. Parameters of object, described in detail in the section 3.2, are stored as tagged values.

Fig. 4 MOM profile in Enterprise Architect (author's research)

4.3 Related work

In this paper I have already mentioned that UML is a common modeling language without any platform or domain dependencies. But exactly these details are crucial and indivisible parts of a model. Also it is the reason, why a lot of extensions have been
Extensions can be realized either as a UML profile, with a tight dependency to UML, or a standalone meta-model. The comparing of both extension mechanisms is described for example in the paper [6]. This article is focused only on UML profiles.

I found some similar works deal with the creation a UML profile for a particular intention. The one of them is the article [7] which described the UML profile for analysis and design applications based on Jakarta Struts framework. This framework was one of the most popular solutions used to develop web applications several years ago. Although the profile contains extensions for five different models (web application model, use case model, flow model, structure model and presentation model) and looks like very ambitious, the extensions are limited to a collection of stereotypes without any advanced elements (tags) affect behavior of framework components. Each stereotype has only a one tag represents a name of component. From my point of view a UML profile should provide more advanced features than simple assigning a name of component. In contrast, every component of the presented MOM profile contains from six to eight properties which have an impact on behavior.

The next similar work is presented in the article [8] focused on modeling of knowledge-based systems. Before the beginning of works on the profile, authors focused on exploring the domain and comparing existing frameworks for modeling knowledge-based systems. I expect that right these activities have a significant impact to an overall very well quality of the profile. In the article authors also present the next step in their research, it should contain the implementation of validation rules using OCL (Object Constraint Language) and testing in real environments. This is a nice idea which I could consider for MOM profile as well.

5. Conclusion

Last few years I have focused on analysis and design of integration applications in different segments (e.g. telecommunication, finance) and have worked with diverse integration platforms which support SOA (Service Oriented Architecture) or BPM. According to my practical experiences I feel that most projects have absolutely same troubles with modeling of an integration layer. Much of analysis and design is realized in a CASE tool and stored in a one repository. However, a middle tier is described only in a high-level and detailed information in a repository is missing. This is due to the lack of detail modeling language, which tries to be universal and usually doesn’t contain platform-specifics properties. Therefore, these properties are filled up during an implementation.

MOM profile is closely associated with a design of integration applications using a type of messaging. It suitably complements common modeling languages such as UML or BPMN. Also it allows to have stored all important features of model in a one central repository. This approach provides an opportunity to generate a completely project documentation including a design of integration tier.

6. Bibliography


[4] IBM WebSphere MQ V7.0.0 Information center [online]. 27th November 2008 [cit. 2010-06-09]. Available from WWW:
<http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0m0/index.jsp>.


<http://www.chris-imble.com/Publ>