REVISTA COLOMBIANA DE COMPUTACIÓN
Volumen 11, número 1
Págs. 56 - 68

A Profile's Design for Parallel Applications Modelling

Daniel Alberto Giulianelli* Claudia Fabiana Pons†
Rocío Andrea Rodríguez* Pablo Martín Vera* Victor Manuel Fernandez*

Fecha de Recibido: 07/12/2009 Fecha de Aprobación: 25/02/2010

Abstract

During the last times hardware progress has reached home computers with technologies that were only used in main frames. Clear examples of this statement are multiple core personal computers. As this new hardware becomes popular it is neccessary to change the way of designing applications in order to be able to use it. UML is a wide general-purpouse modeling language that counts with a graphic vocabulary.

In some cases, when a particular application is going to be modeled, the UML's graphic vocabulary, results too abstract. That's why a specialization of the language is neccessary by means of the adding of new artifacts that allows modeling the special characteristics of the particular domain. This paper shows a profile that specializes UML to facilitate the parellized applications modeling, considering their own characteristics.

Keywords: *Stereotypes, Metamodeling, OCL, Profile, Constraints, UML*

Resumen

El avance del hardware en los últimos tiempos ha traído a máquinas hogareñas tecnologías que sólo eran utilizadas en grandes servidores. Un claro ejemplo de esto son las computadoras con procesadores con múltiples núcleos. Al popularizarse este hardware es necesario un cambio en la forma de diseñar las aplicaciones para poder hacer uso del mismo. UML es un lenguaje de representación de amplio propósito general que cuenta con vocabulario gráfico.

En algunos casos cuando se quiere modelar un tipo de aplicación particular el vocabulario gráfico de UML resulta ser muy reducido. Por esta razón es necesario extender el lenguaje con nuevos artefactos que permitan modelar las características particulares del dominio en cuestión. En este paper se presenta un profile el cual agrega expresividad a UML para modelar aplicaciones paralelizables, teniendo en consideración las características propias de las mismas.

Palabras Claves*:* *Estereotipos, Metamodelado, OCL, Perfil, Restricciones, UML*

[*] La Matanza National University, Florencio Varela 1903, Buenos Aires, Argentina dgilian@unlam.edu.ar , rrodri@unlam.edu.ar , pvera@unlam.edu.ar , vfernandez@unlam.edu.ar
[†] La Plata National University, Calle 50 y 150 La Plata, Buenos Aires, Argentina
cpons@info.unlp.edu.ar

1 Introduction

Nowadays the power of home computers has substantially grown making available to everybody equipment that was only used in big servers due to its high price. The multicore technology included in current microprocessors is a clear example practically, all new equipments include microprocessors with 2 or more cores in the same integrated circut; this technology was reserved only to servers 3 years ago. This new trend rises because of the need of a higher performace through the parallel computing, without the need of increasing the clock speed [5].

The multi-core technology is applied in different devices: desktop computers with 2 to 4 cores, laptops or notebooks with usually 2 cores in the same chip, gaming consoles with up to 9 cores in the same processor (eg: play station 3), or, for example the console X-Box 360 from Microsoft that has 3 processors, each one with 2 cores. Also the integration of these technologies on mobile devices like smart phones is growing.

This new technology opens a new design field, because, if it is necessary to develop high performance software, it cannot be thought as sequential processing like almost all the designers usually do, but it must be thought, from the design the way of parellizing a determined task.

In order to address the problem, it is necessary to count with tools that allow the modeling of these characteristics in one application and hence, it´s necessary having way of identifying: task that can be performed in parallel, exclusively sequential task, communication method among different processors in order to share data, (e.g.: shared memory, services, etc.).

When modeling specific domains some problems arise in the UML (Unified Modeling Language) [2], [13], which make necessary the extension of the language. This language extension will allow the creation of new artifacts dedicated to a specific task or with a determined meaning in the application's domain; hence they will allow the modeling of those features that were not considered within the original UML conception. This extension is defined by building a profile, which is a mechanism used to extend a language in order to express more specific concepts of certain applications' domains.

The following paragraph provides the profile's definition and its objectives according to OMG (Object Management Group) [11]: A ***UML profile*** is a specification that does one or more of the following:

- Identifies a subset of the UML metamodel.

- Specifies "well-formedness rules" beyond those specified by the identified subset of the UML metamodel. "Well-formedness rule" is a term used in the normative UML metamodel specification to describe a set of constraints written in UML's Object Constraint Language (OCL) that contributes to the definition of a metamodel element.
- Specifies "standard elements" beyond those specified by the identified subset of the UML metamodel. "Standard element" is a term used in the UML metamodel specification to describe a standard instance of a UML stereotype, tagged value or constraint.
- Specifies semantics, expressed in natural language, beyond those specified by the identified subset of the UML metamodel.
- Specifies common model elements, expressed in terms of the profile.

In the OMG site it is possible to find different Profiles, for example [11]: CORBA, Data Distribution, Testing, Enterprise Application Integration, System on a Chip.

This work objective is to define a new UML profile, which we call PROCODI, that allows modeling Concurrent and Distributed Processes. The paper is organized in the following way: section 2 describes the steps to build a profile; section 3 shows an example of modeling an application using the proposed tool and, finally, section 4 specifies conclusions and future work.

2 Profile Construction

The definition of this new language (UML based specialization that extends it by adding expressivity to this particular domain), makes necessary to generate a profile [3]. The steps to build the profile are the following:

1. To define the domain entities.
2. To create a stereotype for each entity of the meta modeling, determining the UML's elements that are going to be extended for each stereotype.
3. To define, as profile's elements labeled values, the attributes that appear in the metamodel, including their types' definition and their initial values.
4. For each stereotype a graphic construction is assigned in order to facilitate the modeling, adding expressivity to the new language.

5. To define the well-formedness rules of the domain's specific models.

2.1 Domain's Own Entities

The elements of the UML language extension must be identified in order to be able to determine its meta-modeling. The components that are introduced by PROCODI are the following:

- Semaphores and Monitors: When threads are used, it is very probable that some resources should be shared; hence it is necessary to administrate their access, because only one thread can use the resource at a time.
- Cardinality: This construction is proposed when several threads perform an identical operation.
- Timer: extending cardinality, a construction can be made which represents activities that are performed periodically between a determined time.
- Messages Queues: When using applications with asynchronyc communication, a lot of times, the messages received from different nodes and applications go to a queue, and they remains waiting to be processed until the resource is available. Some examples of this technology implementation are: MQSeries from IBM [7] and Microsoft Message Queuing (MSMQ) [9].
- Web Services: Actually both in GRID [8] and Enterprise environments to achieve the reuse of functionalities among applications, it is possible to expose a determined functionality by means of web services. This fact allows other applications to consume them by a simple http call.

The following elements arise from making adaptations to the preexisting Activities diagram's elements.

- Node: each swim lane of the Activity Diagram will be an application node.
- Thread: A clearer notation is proposed, when mentioning different states for each tread through dotted lines.

Besides, shared memory is very frequently used in parallel applications and concurrent processing. Two specific types of them are detailed as follows:

- Shared memory: a unique and global memory accessible from all the processors.
- Shared - distributed memory: memory is physically

distributed but logically shared.

2.2 Stereotype's generation

Once the metamodeling elements have been identified, the stereotypes' definition for each element that is being extended is performed. It is very important to take into account which UML's metamodeling elements are being extended and to apply a stereotype over them. Some examples are: classes, associations, relationships, operations, attributes, etc. In this way, the stereotype will be applied to a UML metaclass. It can be observed in Table 1, the association among the extension's elements and the metaclasses in which the elements stereotypes are defined. On their turn, the defined stereotypes generate new classes that can be extended too, for example, in Table 1 the following case is shown: the ProTimer is using ProCardinality as a metaclass which is a stereotype that belongs to the profile.

Stereotype	UML Metaclass
ProSwimlane	Synchronization
ProMemory	Class
ProSharedMemory	ProMemory
ProSharedDistributedMemory	ProMemory
ProSharedResourceAccessControl	Class
ProSemaphore	ProSharedResourceAccessControl
ProMonitor	ProSharedResourceAccessControl
ProMessageQueue	MessageEndpoint
ProExtendedMessage	Message
ProMailMessage	ProExtendedMessage
ProSMSMessage	ProExtendedMessage
ProCardinality	Activity
ProTimer	ProCardinality
ProWebService	Activity
ProParalellizableActivity	Activity

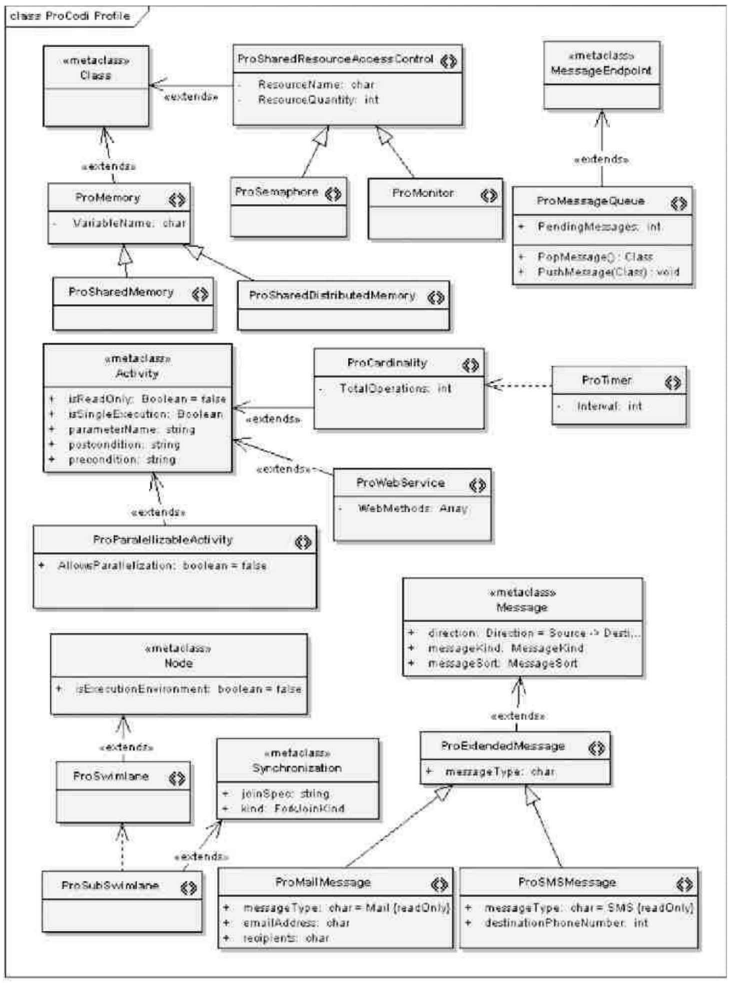
Table 1. Association among Extension Elements and Metaclasses

2.3 Labeled Values Definition

The determination of the profile elements' labeled values is done; these labeled values are attributes of the elements that we are extending. They include their type definition and their possible initial values.

In Figure 1, the stereotypes derived from the metaclasses can be observed: Class, Collaboration, Operation, ActivityParameter, and their labeled values.

Fig 1. Labeled values determination for each profile element.



2.4 Graphic Constructions

For each one of the added stereotypes, in order to extend UML, a graphic construction is built, which will grant more expression to the language. Figure 2 shows the graphic constructions chosed for each one of the stereotypes.

From figure 2 it is possible to see that all the final stereotypes from Table 1 have a graphic construction, except those composity relationship's cases where a graphic construction is not performed for the basic class but for the elements that integrate it. The cases are:

- ProMemory where graphic constructions are created for the defined memory types: Pro Shared Memory y ProShared Distributed Memory
- ProSharedResourceAccessControl where the elements that are modeled are: ProSemaphore y ProMonitor
- ProExtendedMessage: where two graphic constructions are made that allow to distinguish the message type: ProMailMessage y ProSMSMessage

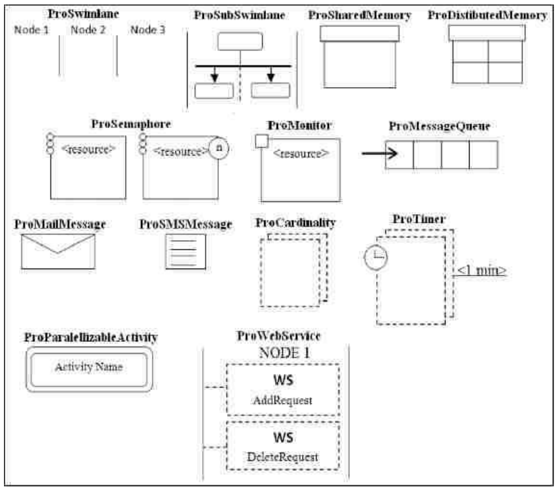


Fig 2. Graphic construction generated for each stereotype

2.5 Constraint Definition

Next, the PROCODI constraints are written. It is possible to perform the task by using a natural language, however OCL (Object Constraint Language) was chosen [12]. The OCL language is adopted like standard language by OMG, to describe UML models' constraints. These constraints can be applied both to a business level objects of a system and to a class or object of a metamodel. The advantage of using a formal language like OCL is the possibility of checking the compliance of the defined constraints through a tool.

These OCL expressions are evaluated into a context; this is the link between an UML diagram's entity and the OCL expression. This context definition specifies the model's entity for which the expression is defined. The reserved word `Context` is used to define the contextual type and when it is used within the OCL expression, the reserved word `self` is used to refer to the contextual instance. For more details over how to write OCL constraints it is advisable to read the paper [1]. To build the profile shown in this paper the following OCL constraints types were required:

- Invariant: The expression must be true for each instance of the classifier at any moment in time.
- Preconditions and Post conditions: The purpose of them is to specify the conditions that must hold before or after the operation executes. The reserved words `pre:` and `post:` are used. Within a post condition, the `@pre` suffix can be used to refer to values at the beginning of the operation.
- Initial and derived values: The model elements' initial values are initialized or their derived values (obtained based on other values).

2.5.1. Invariant Constraint Type

The following pagraphs show the Invariant Constraints of the model, expressed in OCL.

First, the case of the stereotype ProCardinality is written, which represents a set of activities that will replicate in several parallel threads. This set of activities is called operation and so the final quantity of performed operations will be greater than 1, because they will be executed in several threads.

```
Contex ProCardinality
inv : self.TotalOperations>1
```

When defining a frequency time with which an operations set will be executed in several threads, it´s modeled with the graphic construction ProTimer (see Picture 2). For this stereotype, the constraint that the time interval set must be greater than 0, is specified.

```
Contex ProTimer
inv : self.Interval>0
```

Every defined Web Service must have at least one established method, which is express in OCL in the following way:

```
Contex ProWebService
inv : self.WebMethods -> size()>0
```

It also explicits that both the semaphores and monitors must have indicated the quantity of resources to share at least equal to 1.

```
Contex ProSharedResourceAccessControl
inv : self.ResourceQuantity >0
```

Finally the type of message that cannot be modified at any instance of those classes is defined for the two derived sub classes of ProExtendedMessage.

```
Contex ProMailMessage
inv : messageType=”Mail”
Contex ProMailMessage
inv : messageType=”SMS”
```

2.5.2. Pre/Post Conditions Contraint Type

In the following example, the messages' queue restriction is written in OCL: when the message counter is greater than 0, and a pending message is taken to process it, the message counter will take the previous value minus 1 which is the message removed from the queue.

```
Contex ProMessageQueue::PopMessage():Class
pre: self.PendingMessages>0
post: self.PendingMessages=self.PendingMessages@pre -1
```

The following case shows how to add a new message to the queue.

```
Contex ProMessageQueue::PushMessage(message:Class):Void
pre: messages->notEmpty()
post:self.PendingMessages=self.PendingMessages@pre+1
```

2.5.3. Initial and Derive Values Constraint Type

It was necessary to write a constraint to fix the initial default value for the attribute “AllowsParallelization” (parallelizable) which belongs to the class ProActivity, that implies that none activity will be parallelizable except it is indicated.

```
Contex ProActivity::AllowsParallelization: :Boolean
init:  -> false
```

3 An Application Modelling

Figure 3 shows an example of an application modeling that requires the parallel processing execution. In this application, a reservation process of a shared resource allocated in a determinate node arises. The resources reservations are made before the moment when the resource has to be used and each client requests the time range when he will need the resource. The requests of the different clients, previously registered, made all day long, are taken for that resource reservation. At the end of the days, the turn's assignment for the use of the concerned resource is made among all the received requests. If there is a time overlapping, the system will calculate an independent priority for each client. This priority is determined based on previously stored parameters like: last reservation date, reservation compliance, use' purpose importance, etc. Once it has been determined to which of the clients the resource will be assigned, the system automatically sends a notification mail with an access password to the shared resource and the reservation acceptance's confirmation.

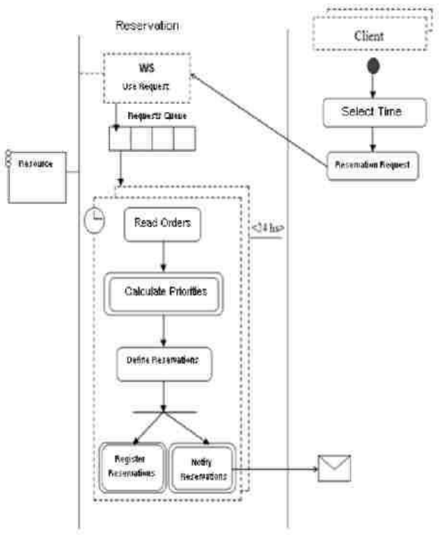


Fig. 3. Reservation Process Modeling using PROCODI

4 Related Work

The necessity of extending the language is shared with other colleagues. Some of them have generated particular domain's profiles others have taken previous steps setting out which would be the necessary adaptations.

- The article [4] specializes in the concurrence modeling. This article, the same as ours, show the necessity of using semaphores and monitors. They don´t build graphic constructions for modeling but they generate classes and by means of the class diagram they represent these resources. Another contact point with our work is the use of OCL as a formal restriction language.

- The profile showed in OMG [10], allows to model characteristics of real time systems. The authors generate graphic constructions for the proposed stereotypes. A lot of the metaclasses adds elements that respond to low level characteristics, some of them, nowadays, are performed by the operative system being transparent to the user. N this domain the following graphic construction were added: Timer, Shared eAccess, Messages, etc. If it want to be showed if the access is made by a semaphore or a monitor, it must be specified without graphic construction. The restrictons' definition is mada by natural language instead of a formal specification like OCL.

- The work [6] shows the real time systems modelling. The authors use the capsule concept, where the objects groupped into the same capsule match with the objects that are executed in the same thread, similar to the concep of substreets presented in this work.

- The article [14] streets are used to represent each one of the process, they add stereotypes without the definition of graphic construction for each one of them. They neither apply restrictions nor generate a language formal extension, but, as the article's title shows, they plan to adapt UML to this particular domain.

5 Conclusions

This paper shows the necessity of extending the UML vocabulary to model the specific characteristics of a determined domain. As it could be watch, through the present work, creating an extension over UML implies following a set of steps to formally define the new language.

This language relies on the UML conformation basis; it doesn´t change UML's semantics but it specializes UML in order to model the domain characteristics. The exposed mechanism set has allowed the definition of a Profile showing the extensions performed by stereotypes, their graphic constructions, and the necessary constraint written in OCL.

There are two main advantages in the use of OCL:

- To avoid the ambiguity that may arise when defining constraints in natural language;
- The possibility of analyze and validate UML models with OCL expressions using tools in the constraints definitions.

UML could have been extended without new graphic constructions, simply adding labels over the already existing graphic constructions, but UML has been created like a graphic language and this mechanism would take away readability to the language.

As future work, we propose to build this same extension taking DSL (Domain Specific Language) [8] as a basis, to be able to compare both approaches.

6 Acknowledgements

We thank Dr. Leon Welicki (Salamanca Pontificial University, campus Madrid and Reedmon Microsoft, USA), and CC. Artemisa Trigueros (La Matanza National University, Argentina) because of their collaboration with the present work.

References

[1] Becker V., Pons C. 2003. Definición Formal de la Semántica de UML-OCL a través de su traducción a OBJECT-Z. Universidad Nacional de La Plata, Facultad de Informática, LIFIA-Laboratorio de Investigación y Formación en Informática Avanzada, pp 2-6.

[2] Booch G, Rumbaugh J y Jacobson I. 2005. Unified Modeling Language User Guide. Addison Wesley, 2nd Edition.

[3] Fuentes L. y Vallecillo A. 2004. Una Introducción a los Perfiles UML. Depto. de Lenguajes y Ciencias de la Computación, Universidad de Málaga Campus de Teatinos. España.

[4] Goñi A., Eterovic Y., 2004. Building Precise UML Contruscts to Model concurrency Using OCL, Seventh Intarnational Conference on the Unified Modeling Language and its applications.

[5] Grama A., Karypis G., Kumar V., Gupta A. 2003. Introduction to

Parallel Computing. Pearson. Addison Wesley. 2nd Edition.

[6] Gu Z., Shin K., 2004. Synthesis of Real-Time Implementation from UML-RTModels. Real-Time Computing Laboratory. Department of Electrical Engineering and Computer Science. University of Michigan., USA. URL: http://kabru.eecs.umich.edu/aires/paper/gu_modes04.pdf

[7] IBM, MQSeries. URL: http://www-306.ibm.com/software/integration/wmq/

[8] Kelly S., Tolvanen J., 2008. Domain-Specific Modeling. Wiley-IEEE Computer Society Pr

[9] Magoules F., Pan J., Tan K., and Kumar. 2009 A. Introduction to Grid Computing. Publisher CRC.

[10] Microsoft, Message Queuing (2003). URL: http://www.microsoft.com/windowsserver2003/technologies/msmq/default.mspx

[11] OMG, A UML Profile for MARTE (Modeling and Analysis of Real Time and Enbedded Systems. URL: http://www.omgmarte.com.org/

[12] OMG. Catalog of UML Profile of Specification. 2009 . URL: http://www.omg.org/technology/documents/profile_catalog.htm

[13] OMG, OCL Specification, Version 2.0, (2006) URL: http://www.omg.org/spec/OCL/2.0/

[14] OMG, Unified Modeling Language, Infrastructure, Version 2.1.2 (2007) URL: http://www.omg.org/docs/formal/07-11-04.pdf

[15] Pllana S. y Fahringer T., 2002. Institute for software siencie Austria.Vienna. On Customizing the UML for Modeling Performance-Oriented Applications. URL: http://portal.acm.org/citation.cfm?id=719627