Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A framework for an integrated unified modeling language*

Mohammad ALSHAYEB‡, Nasser KHASHAN, Sajjad MAHMOOD
(*Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia*)
E-mail: alshayeb@kfupm.edu.sa; khashan@live.com; smahmood@kfupm.edu.sa

Received Mar. 23, 2015; Revision accepted July 22, 2015; Crosschecked Jan. 20, 2016

Abstract: The unified modeling language (UML) is one of the most commonly used modeling languages in the software industry. It simplifies the complex process of design by providing a set of graphical notations, which helps express the object-oriented analysis and design of software projects. Although UML is applicable to different types of systems, domains, methods, and processes, it cannot express certain problem domain needs. Therefore, many extensions to UML have been proposed. In this paper, we propose a framework for integrating the UML extensions and then use the framework to propose an integrated unified modeling language-graphical (iUML-g) form. iUML-g integrates the existing UML extensions into one integrated form. This includes an integrated diagram for UML class, sequence, and use case diagrams. The proposed approach is evaluated using a case study. The proposed iUML-g is capable of modeling systems that use different domains.

Key words: Unified modeling language (UML), Integration, Modeling, System analysis and design
http://dx.doi.org/10.1631/FITEE.1500094 **CLC number:** TP311

1 Introduction

The unified modeling language (UML) (Booch *et al*., 2005) is a modeling language used to specify, visualize, construct, and document aspects of the system-development process. Although UML provides a set of graphical notations, which helps in expressing the object-oriented analysis and design of software projects, some software engineers found UML unable to cover some problem domains. For this reason, UML allows its users to customize it to address the desired problem domains. This is done by UML extension mechanisms that enable UML to better adapt to a variety of different domains. These mechanisms allow the user to leverage the existing UML specifications to the desired level, thereby making modeling easier. Atkinson *et al*. (2015) proposed a modeling framework that was best able to support the extension scenarios.

There are two types of UML extension mechanisms, UML lightweight extension and UML heavyweight extension. UML lightweight extension involves using profiles (Magureanu *et al*., 2013; Hsu *et al*., 2014; Lara *et al*., 2014; Boulil *et al*., 2015). A UML profile defines limited extensions to the metamodel elements. It uses three main constructs: stereotypes, tag definitions, and constraints. This type of UML extension provides a simple and straightforward mechanism for customizing existing UML modeling elements to a particular domain. It does not change the UML behavior, but it can add to or modify the UML structure. The second type is UML heavyweight extension (Zubcoff *et al*., 2009; Génova *et al*., 2014), which involves the reuse technique of the UML package. It also involves two steps: selecting the desired modeling elements that one wants to extend, and merging them with the elements from the targeted problem domain. It can customize UML behavior and operations, but its development is difficult and costly.

In general, UML extensions add new terminologies and properties and define new semantics to

‡ Corresponding author
* Project supported by the King Fahd University of Petroleum and Minerals, Saudi Arabia (No. IN100046)
ORCID: Mohammad ALSHAYEB, http://orcid.org/0000-0001-7950-0099

make the language suitable to a specific problem domain. The problem is that, after extending UML, it becomes suitable only for a specific domain, which may make it unusable for other domains even if they differ only in small details. In this paper, we propose a framework for integrating the UML extensions, and then use the framework to integrate the available UML extensions in the literature to form an integrated UML-graphical (iUML-g) form. The motivation for this work is to reduce the time and effort invested during modeling the targeted system using UML extensions. iUML-g tends to save time and effort when it comes to modeling, since it provides one integrated form for all required problem domains. iUML-g also provides the designers with a flexible way to model the targeted systems. iUML-g gives one broad set of graphical concepts to model different domains at the same time.

2 Literature review

This section surveys the literature on the extensions of class, sequence, and use case diagrams. These three diagrams are the most commonly used representatives for three distinctive views of the modeled system. The class diagram depicts the system's structure, the sequence diagram represents the interactions between the system's objects, and the use case diagram describes the provided functionality of the system.

2.1 Class diagram

Fontoura *et al*. (2000) proposed a new profile called UML-F, which describes how to represent framework variation points in UML diagrams to describe the structure and behavior of these variation points. Byeon *et al*. (2004) used a diagrammatic tool called 'stereotype creator' to create iconic stereotypes to model the global navigation satellite system (GNSS) application. The main elements of geo-referenced classes are a graphical representation with a symbolistic icon and an iconic notation to indicate the geographic type, class name, attributes, and operations.

Dong (2002) presented notations to represent individual and composed design patterns. The author believed that identifying the design patterns is extremely difficult, especially when they are composed, because some pattern-related information may become truncated or even lost when using traditional UML diagrams. Dong (2002) showed a number of annotations for design patterns, including Venn-diagram-style pattern annotation, dotted bounding pattern annotation, UML collaboration notation, pattern role annotations, stereotype annotations, and tagged pattern annotation. Dou *et al*. (2013) reused the UML meta-model definition and proposed a metamodeling approach for pattern specification.

Sanada and Adams (2002) defined a new UML profile to model design patterns and frameworks in design class diagrams. This work distinguishes between design class diagrams, detailed design class diagrams, and design pattern class diagrams. Sanada and Adams (2002) also added stereotypes and tags to model frameworks. Peterson *et al*. (2006) used a UML class diagram to represent an automated teller machine (ATM) model integrated with UMLpac for possible security considerations. Without extending UML, it would be challenging for UML to model the secured health care system using regular notations and other modeling elements. Mahmood and Lai (2013) presented an extension to UML called RE-UML to support the phases of requirements analysis and assessment process (RAAP). RE-UML extends the UML class diagram with two specialized classes: RClass to specify stakeholder requirements, and CClass to specify component features. Jantan *et al*. (2008) proposed a hypermedia design method called ComHDM, which is a UML profile. The authors proposed modeling elements to model the conceptual, navigational, and user interface artifacts of web hypermedia applications. Fernández-Medina *et al*. (2007) addressed the confidentiality problems of data warehouses by specifying security constraints in the conceptual multidimensional database model to design secure data warehouses. Cunha *et al*. (2015) proposed a model transformation from alloy to UML class diagrams annotated with object constraint language (OCL).

2.2 Sequence diagram

Zhou *et al*. (2008) made three contributions: first, they proposed a UML extension profile for aspect-oriented modeling; second, they built a framework for UML; and finally, they presented a way to model the dynamic behaviors that occur in aspect-oriented software. Their main objective was to propose an architecture for aspect-oriented modeling

and address the separation of concerns properly. Hausmann *et al*. (2001) specified the operational semantics of UML behavioral diagrams. They extended the sequence diagram by introducing a new modeling construct (synchronization).

Xie *et al*. (2007) proposed synchronization adorned UML (saUML) sequence diagram notation to highlight aspects of thread interactions. Their main objective was to investigate whether the proposed graphical notation made it easier to understand concurrent executions and concurrency concepts as opposed to purely textual representations. They found that the proposed representation was beneficial compared to a text-only presentation.

Seemann and von Gudenberg (1999) defined a textual language UMLscript-RT to describe UML sequence diagrams, adding an explicit loop and alternative statements for the simulation of real-time systems. da Silva and de Lucena (2004) proposed a multi-agent system modeling language (MAS-ML) that extended the UML class and sequence diagrams. For the sequence diagram, they proposed three new stereotypes (<<role commitment>>, <<role cancel>>, and <<role change>>).

Saleh and El-Morr (2004) proposed an extension to UML (M-UML) that covered all aspects of mobility at the various views and diagrams of UML. For sequence diagrams, they proposed a new stereotype <<localized>> to show when mobile interactions need not be co-located. Fontoura *et al*. (2000) proposed the UML-F, which allows the explicit representation of framework variation points. They extended both the class and sequence diagrams. For the sequence diagram, they added the tag {optional} to indicate interactions that are not mandatory.

Fei and Yan (2008) analyzed a real application called SPAERIS using a UML extension called Agent UML. SPAERIS is an application used to monitor and control a ship's security. They used Agent UML to design a distributed management information system.

Cruz-Lemus *et al*. (2011) presented a number of experiments to investigate whether the use of stereotypes improves the comprehension of UML sequence diagrams.

2.3 Use case diagram

Dong *et al*. (2002) proposed an extension to UML to address a distributed system. Their UML extension changes the use case diagram to be active and multilevel for requirement engineering of a distributed system. Djemaa *et al*. (2006) presented web-adaptive UML (WA-UML), which is a UML profile to model adaptive web applications. This profile adds labels and notations to UML diagrams in order to express UML more effectively.

Chung and Supakkul (2006) proposed a UML extension to represent the nonfunctional requirements with functional requirements in the use case model. Stein *et al*. (2002) extended UML to present aspects. Misbhauddin and Alshayeb (2015) provided an extension to the UML use case metamodel to facilitate model analysis and interchange. Table 1 summarizes all the discussed extensions.

3 Extension integration

iUML-g provides a flexible method for combining different UML extensions. It provides a process to integrate available or new UML extensions. In software systems that use different domain applications, a designer may need to combine the notation of more than one UML extension. The designer will need to consider the overlap and conflicts between the targeted extensions. iUML-g provides a set of graphical notations, which removes the overlap and conflict between the integrated extensions. The iUML-g integration process that integrates the available UML extensions is discussed in the following subsections.

3.1 Integration process

The integration process is applied to UML extensions that provide graphical symbols. The process starts by creating a graphical library that contains the graphical symbols themselves and their descriptions. Extensions that do not cause any conflict and that keep the original intent of the symbols clear are then integrated. In other words, the final symbol must deliver the idea behind it without any confusion. The following process explains the integration of graphical symbols:

1. Creation of a library: Create a library for the graphical symbols. The library will contain the graphical symbols themselves and their descriptions.

Table 1 Summary of lightweight UML extensions

Reference	Domain	Purpose of extension	Diagram
Fontoura *et al*. (2000)	Object-oriented frameworks	To model variation points in UML diagrams	Class & sequence
Byeon (2004)	Global navigation satellite system	To provide notational help for accurate calculations of real-world geographical entities	Class
Dong (2002)	Design patterns compositions	To represent design patterns in the application and composition of design patterns and maintain pattern-related information	Class
Sanada and Adams (2002)	Design patterns	To model design patterns and frameworks in design class diagrams (DCDs)	Class
Peterson *et al*. (2006)	Security	To incorporate security techniques into software class design	Class
Mahmood and Lai (2013)	Component-based software system	To specify satisfaction and risk assessment to evaluate customer demands against component features	Class
Jantan *et al*. (2008)	Web hypermedia applications	To model complicated design issues	Class & activity
Fernández-Medina *et al*. (2007)	Data warehouses	To address confidentiality problems and set security constraints in the conceptual modeling of data warehouses	Class
Zhou *et al*. (2008)	Aspect-oriented modeling (AOM)	To model the functional crosscutting concerns and integrate the AOM architecture	Sequence
Hausmann *et al*. (2001)	UML semantics specification	To integrate extensions' specific semantics with UML semantics	Sequence
Xie *et al*. (2007)	Multithreading and concurrency	To highlight aspects of thread interactions	Sequence
Seemann and von Gudenberg (1999)	Real-time	To define a textual language UMLscript-RT to describe the sequence diagrams	Sequence
da Silva and de Lucena (2004)	Agents	A multi-agent system modeling language (MAS-ML)	Sequence
Saleh and El-Morr (2004)	Mobile agent-based software systems	The extension covers all aspects of mobility at the various views and diagrams of UML	Sequence
Fontoura *et al*. (2000)	Frameworks	UML-F that allows the explicit representation of framework variation points	Sequence
Hausmann *et al*. (2001)	Extensible semantics	To specify the operational semantics of UML behavioral diagrams	Sequence
Dong *et al*. (2002)	Distributed systems	To change the use case diagram to multilevel for requirement engineering of a distributed system	Use case
Djemaa *et al*. (2006)	Adaptive web application	To model adaptive web applications (AWA)	Use case
Fei and Yan (2008)	Agent UML	To enhance the analysis and design of an agent system	Use case
Chung and Supakkul (2006)	Requirements	To represent the nonfunctional requirements with the functional requirements	Use case
Stein *et al*. (2002)	Aspects	To present aspects	Use case

2. Case A (Combination): For each type of UML diagram, combine possible graphical symbols that cause no graphical conflicts, but make sure that the final symbol still represents its intended goal.

3. Case B (Conflict): In case of a graphical conflict, insert each graphical symbol on its own into the library.

3.2 Inclusion and exclusion criteria

We define the inclusion/exclusion criteria; only extensions that meet our inclusion criteria are included in iUML-g and the others are excluded. The inclusion criteria are as follows:

1. UML lightweight extensions;

2. Extensions that provide graphical notation/icons for the notation;

3. UML class, sequence, and use case diagram extensions only;

4. UML domain-specific extensions that can be combined with other same domain-specific extensions, preferably working on different areas of the extension but at the same level;

5. UML domain-specific extensions that can be combined with the other different domain-specific extensions, preferably general extensions;

6. When two UML extensions focus on one particular area and on one type of UML diagram, combine them together or choose the more general one.

The exclusion criteria are as follows:

1. UML activity, component, state chart, interaction diagrams;

2. UML heavyweight extensions that manipulate the UML meta-model by editing or deleting UML packages;

3. Theoretical and algorithmic UML extensions;

4. Profiles.

3.3 Applying the integration process

In this subsection, the integration process mentioned above is applied to three UML diagrams: class, sequence, and use case. In each subsection, a step-by-step explanation of the integration process is shown.

3.3.1 Integration of graphical symbols

This subsection addresses the application of the integration process on the UML class, sequence, and use case diagram graphical extensions. This process has three steps: creation of a library, integration, and conflict handling. Each UML diagram will be subjected to these steps, and the results will be shown as the process is applied.

1. Class diagram

Step 1: Creation of a library

In this process of graphical integration, a library is created to include the proposed graphical extensions. All the graphical symbols are inserted along with their descriptions. The idea behind having such a library is to have a graphical database for iUML-g. Such a database lists all the symbols and their descriptions, plus their original source. The description column informs the user of the intended objective of the symbol. Table 2 shows the created library for UML class diagram graphical extensions.

Step 2: Case A (Combination)

If some of the already existing symbols in the library can be combined together with other existing symbols, combine them into one symbol and add that symbol to the library. Table 3 shows the integrated graphical symbols.

Step 3: Case B (Conflict)

If a graphical conflict occurs between two or more extensions, these extensions should be inserted individually in the library. In the process of integrating a UML class diagram, no graphical extensions are found to have a conflict.

2. Sequence diagram

Step 1: Creation of a library

Table 4 shows the created library for UML sequence diagram graphical extensions.

Step 2: Case A (Combination)

The result of this step is one integrated symbol. Table 5 shows this symbol.

Step 3: Case B (Conflict)

No conflict is found in the sequence diagrams extensions.

3. Use case diagram

Step 1: Creation of a library

Table 6 shows the created library for UML use case diagram graphical extensions.

Step 2: Case A (Combination)

The result of this step is one integrated symbol. Table 7 shows this symbol.

Step 3: Case B (Conflict)

One conflict occurs during the attempt to integrate three graphical extensions. Table 8 shows the three symbols that cannot be integrated.

The goal behind integrating these functionalities is to have one abstract use case. However, during the creation of the diagram, the abstract use case makes the diagram confusing because every time there is a need for a specific functionality, one has to refer to the abstract use case. Therefore, it is better to have three independent functionalities where each one presents a different type of information.

3.4 Qualitative assessment

In this subsection, we present qualitative analysis of the feedback received from software engineers and system analysts, with industrial experience, on using iUML-g. The participants were provided complete technical details of iUML-g to implement it in their own projects. A total of nine professionals participated in the study.

Table 2 Library of the proposed graphical symbols (class diagram)

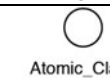
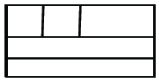
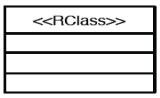
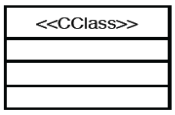
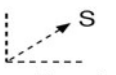
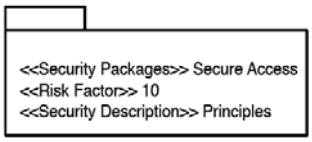
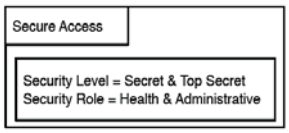
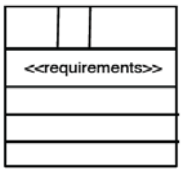
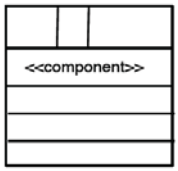
Modeling element	Source	Meaning of the symbol
 Atomic_Class	Jantan *et al*. (2008)	A single process
 Database_Class	Jantan *et al*. (2008)	A database in the class diagram design
Database_Link	Jantan *et al*. (2008)	The information and data operations (such as query, lookup, and entry) that are involved with the database
 Interaction_Class	Jantan *et al*. (2008)	Complex interaction between users and web applications
 Navigational_Class	Jantan *et al*. (2008)	Hyperlinks in the class diagram design
 NonAtomic_Class	Jantan *et al*. (2008)	Predefined and complex processes
 Process_Class	Jantan *et al*. (2008)	The user's action to perform activities
{variable}	Fontoura *et al*. (2000)	The implemented methods during the framework instantiation
{appl-class}	Fontoura *et al*. (2000)	Classes that are defined as framework instances
{extensible}	Fontoura *et al*. (2000)	The extensibility of class functionality
{static}	Fontoura *et al*. (2000)	Variation points of non-runtime instantiation
{dynamic}	Fontoura *et al*. (2000)	Variation points of runtime instantiation
{incomplete}	Fontoura *et al*. (2000)	The possibility of adding new subclasses
{forAllNewMethods}	Fontoura *et al*. (2000)	Indicating that the OCL constraint must be met by the introduced methods
{optional}	Fontoura *et al*. (2000)	Optional event
{final}	Dong (2002); Sanada and Adams (2002)	Indicating that the final class has no decedent classes (leaves)
	Byeon *et al*. (2004)	The geo-referenced class is used to represent the class icon with the aid of graphical notations. The main elements of geo-referenced classes are a graphical representation with a symbolistic icon, an iconic notation to indicate the geographic type, class name, attributes, and operations
	Mahmood and Lai (2013)	RClass is used to represent stakeholder requirements and is divided into four sections: first, stereotyped requirement text, name of the class, and abstraction level to differentiate the requirement level; second, the objective of the RClass; third, scenario, which is the set of interactions necessary to achieve the objective; fourth, rank of the RClass
	Mahmood and Lai (2013)	CClass is used to represent component features and is divided into three sections: first, stereotyped component text and name of the class; second, the functionality provided by the component; third, the dependency on elements and their relationships
	Fernández-Medina *et al*. (2007)	Security information and constraints
	Fernández-Medina *et al*. (2007)	Dimensions within a multidimensional model
	Fernández-Medina *et al*. (2007)	Facts within a multidimensional model
	Fernández-Medina *et al*. (2007)	Dimension hierarchy levels within a multidimensional model

Table 3 Integrated graphical extensions

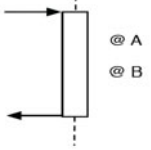
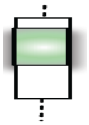
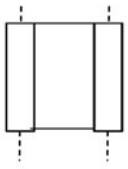
Modeling element	Source	Meaning of the symbol	Method of combination
	Peterson *et al*. (2006); Fernández-Medina *et al*. (2007)	The security package will be inserted into the class diagram and will be attached to the classes that need to be protected from security attacks. Each security package has three attributes: risk factor, which calculates the probability of the security attack; security tile, which protects the main parts of a system; security descriptor, which protects specific parts of the system	The design of the security package was adopted from Peterson *et al*. (2006), while the security information was suggested by Fernández-Medina *et al*. (2007)
	Peterson *et al*. (2006); Fernández-Medina *et al*. (2007)	A security tile that protects the main parts of the system. It mostly contains tagged values specified by security analysts and can be attached to security packages to cover more security concerns	Same as above
	Byeon *et al*. (2004); Mahmood and Lai (2013)	The new main elements of the class are three vertical compartments to indicate symbolistic icons, iconic notations, and class name, and <<requirements>> to specify stakeholder requirements. It will be used to represent requirements with the aid of graphical notations	The three vertical compartments that will contain some graphical and textual information were suggested by Byeon *et al*. (2004). The requirements stereotype and the other requirements-related information were proposed by Mahmood and Lai (2013)
	Byeon *et al*. (2004); Mahmood and Lai (2013)	The new main elements of the class are three vertical compartments to indicate symbolistic icons, iconic notations, and class name, and <<component>> to specify stakeholder requirements. It will be used to represent requirements with the aid of graphical notations	The three vertical compartments that will contain some graphical and textual information were suggested by Byeon *et al*. (2004). The component stereotype and the other requirements-related information were proposed by Mahmood and Lai (2013)

Qualitative data were collected by conducting interviews with the participants. Their experiences were documented using mainly two questions encompassing the advantages and difficulties associated with applying the proposed iUML-g notation. The interviews were kicked off with the following question: "Does iUML-g provide a broad set of graphical concepts to model different domains?" Next, participants were asked to answer the following question: "As compared to UML, do you find iUML-g more capable of modeling systems that involve more than one application domain?" We used follow-up questions to clarify and gather more details about the strengths and suggested improvements mentioned by the participants. The interview participants were also asked to rate each question as either 'strongly agree', 'agree', 'neutral', 'disagree', or 'strongly disagree'.

As shown later in Table 9, the overall average for all questions is above 3.5 on a scale of 4. The interview data indicate that seven out of the nine participants strongly agreed that iUML-g provided a set of graphical concepts to model different domains. In response to the second question, 66% participants strongly agreed that iUML-g was more capable of modeling systems that involve more than one application domain. More than 88% participants either strongly agreed or agreed that iUML-g had a short learning curve. Similarly, all the participants either strongly agreed or agreed that the tool support facilitated using iUML-g in practice.

The participants did not indicate any major disadvantages in applying the iUML-g in modeling software that involved more than one application domain. Furthermore, three participants suggested the incorporation of extensions to other UML diagrams (e.g., activity and collaboration diagrams). We agreed with these participants, and had incorporated their suggestions in our plan for future work.

Table 4 Library of the proposed graphical symbols (sequence diagram)

Modeling element	Source	Meaning of the symbol
{variable}	Fontoura *et al*. (2000)	The methods that must be implemented during the framework instantiation
{appl-class}	Fontoura *et al*. (2000)	Classes that are defined and used as framework instances
{extensible}	Fontoura *et al*. (2000)	The extensibility of class functionality
{static}	Fontoura *et al*. (2000)	Variation points of non-runtime instantiation
{dynamic}	Fontoura *et al*. (2000)	Variation points of runtime instantiation
{incomplete}	Fontoura *et al*. (2000)	The possibility of adding new subclasses
{forAllNewMethods}	Fontoura *et al*. (2000)	Indicating that the OCL constraint is meant to hold for all newly introduced methods
{optional}	Fontoura *et al*. (2000)	Indicating that a given event is optional
{final}	Dong (2002); Sanada and Adams (2002)	Indicating that the final class has no decedent classes (leaves)
	Zhou *et al*. (2008)	Crosscutting bar to indicate join points between two events
	Hausmann *et al*. (2001)	Synchronization bold bars to be placed between activations, meaning that the activities must start and end at the same time
	Xie *et al*. (2007)	Indicating the threads and colors to distinguish between running, ready, or suspended threads
	Seemann and von Gudenberg (1999)	Loops and constraints in textual format
<<role cancel>>	da Silva and de Lucena (2004)	An agent canceling its role
<<role commitment>>	da Silva and de Lucena (2004)	An agent committing to a role
<<role change>>	da Silva and de Lucena (2004)	An agent changing its role
<<localized>>	Saleh and El-Morr (2004)	Indicating that mobile interactions need not be co-located
 crosscutting messages	Zhou *et al*. (2008)	Crosscutting messages
 synchronization	Hausmann *et al*. (2001)	Synchronization in the modeling construct

4 Tool support

All of the UML extensions' modeling elements were modeled and integrated by a special diagram editor tool called Dia (Hsia *et al*., 1995). Dia is a free software that allows the user to create diagrams with the aid of a wide selection of modeling elements. Elements come from domains such as Cisco, Database, Electric, Flow Chart, UML, and others. The Dia tool is known for its simple and easy-to-use environment. Dia makes it easy to control and manage the drawn elements of diagrams through the provided properties attached to each element. The drawing mechanism in Dia is as easy as using the Paint tool found in Microsoft Windows releases. It is easy to handle and flexible.

Table 5 Integrated graphical extension

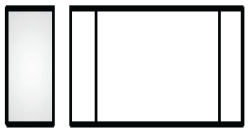
Modeling element	Source	Meaning of the symbol	Method of integration
	Hausmann *et al*. (2001); Zhou *et al*. (2008)	The crosscutting bar indicates join points that must start and end at the same time	The crosscutting bar was suggested by Zhou *et al*. (2008) to show the join points between two events. Hausmann *et al*. (2001) proposed the other graphical symbol to enforce synchronization between two activities. Both symbols focus on the start time of the activity, and hence the final integrated symbol indicates synchronizing join points

Table 6 Library of the proposed graphical symbols (use case diagram)

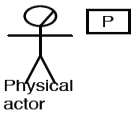
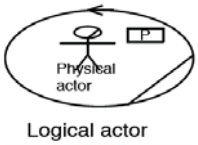
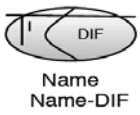
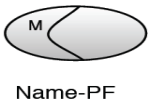
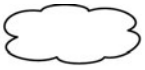
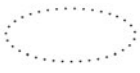
Modeling element	Source	Meaning of the symbol
	Fei and Yan (2008)	Agents
	Djemaa *et al*. (2006)	The human user who visits the web application
	Djemaa *et al*. (2006)	The role played by a human user (physical actor) to maintain the web application
	Djemaa *et al*. (2006)	The hardware aspect of the system, whether it is a computer system, device hardware, or web service
	Djemaa *et al*. (2006)	DIF (dynamic informational functionality) is used to represent a dynamic web page
	Djemaa *et al*. (2006)	SIF (static informational functionality) is used to represent a static web page
	Djemaa *et al*. (2006)	PF (profession functionality) is used to represent a dynamic web page using update request
	Chung and Supakkul (2006)	Nonfunctional requirements
	Chung and Supakkul (2006)	Operationalizing nonfunctional requirements
	Chung and Supakkul (2006)	Claiming nonfunctional requirements
<<refine>>	Stein *et al*. (2002)	Refined aspects

Table 7 Integrated graphical extension

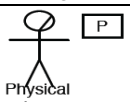
Modeling element	Source	Meaning of the symbol	Method of integration
	Djemaa *et al*. (2006); Fei and Yan (2008)	The human user who visits the web application, or agents in agent-oriented systems	The human user symbol suggested by Djemaa *et al*. (2006) is more general, and hence can represent agents in agent-oriented systems

Table 8 The three extended functionalities proposed by Djemaa *et al*. (2006)

Modeling element	Meaning of the symbol
	DIF (dynamic informational functionality) is used to represent a dynamic web page
	SIF (static informational functionality) is used to represent a static web page
	PF (profession functionality) is used to represent a dynamic web page using update request

Table 9 Study qualitative data (9 participants, on a scale of 4)

Question	Number of participants					Average
	Strongly agree (4)	Agree (3)	Neutral (2)	Disagree (1)	Strongly disagree (0)	
Does iUML-g provide a broad set of graphical concepts to model different domains?	7	2	0	0	0	3.78
As compared to UML, do you find iUML-g more capable of modeling systems that involve more than one application domain?	6	3	0	0	0	3.67
Does iUML-g have a short learning curve?	6	2	1	0	0	3.56
Does iUML-g tool support facilitate using iUML-g in practice?	8	1	0	0	0	3.89

Using Dia, the user can insert text, control the size of the drawn elements, and enter properties for such elements. What makes Dia more interesting than the other diagram editor tools is its ability to control and specify the diagram elements. Each element in the diagram has properties. For example, the element 'Class' has properties such as name, attributes, and operations, which can be specified by the user by double-clicking the element in the diagram and then entering the desired information. The user can also choose if he or she wants the class to be abstract or the class's attributes to be visible or not. Another feature is the ability to create a stereotype for the user's class, which makes the procedure of extending the diagram easier, becoming just a simple text-entering procedure.

Another extraordinary feature found in Dia is the option to create a sheet of modeling elements, i.e., drawing elements from scratch and saving them in a special library or sheet. This sheet can be listed in the main menu of sheets and can be easily used.

In this work, Dia was used to help in creating integrated graphical extensions. The need was for a diagram editing software that provides flexible editing tools, which makes the process of integrating graphical symbols easy and straightforward. In addition, there was a need for software like Dia to store the final integrated symbols in a ready-to-use library and, as mentioned earlier, Dia provides a way to store the created symbols in sheets. After saving the symbols in a sheet, they will be easily selected and used during the creation of diagrams.

An iUML-g sheet was created using Dia (Hsia *et al*., 1995). This sheet contains modeling elements from the collected UML extensions, plus the integrated ones. Fig. 1 shows the iUML-g sheet.

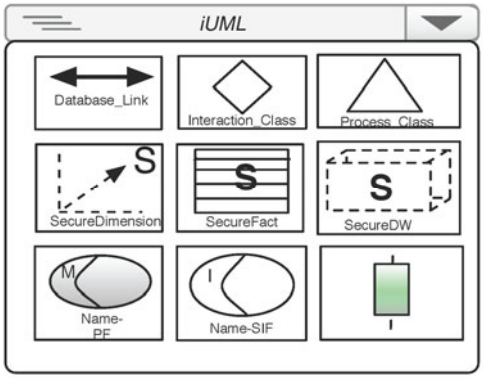


Fig. 1 iUML-g sheet (a subset of iUML symbols)

An example of the created modeling elements is the three integrated classes proposed by Byeon *et al*. (2004) and Fernández-Medina *et al*. (2007), as shown in Fig. 2. Fernández-Medina *et al*. (2007) proposed security constraints such as security levels and roles to be placed on the elements of a hospital system, and Byeon *et al*. (2004) suggested that the class graphic format can be vertically divided to include helpful graphical iconic notations. The results are integrated classes, like the ones shown in Fig. 3.

The class diagram shown in Fig. 3 was created using Dia. Three classes were created: Student, GPA, and Registrar. Class 'Student' is a component class that satisfies the requirements of class GPA, a requirement class. The three classes (symbols) in this example are iUML-g symbols. The way the classes are drawn is by integrating two extensions: those of Mahmood and Lai (2013) and Byeon *et al*. (2004).

5 Case study

This section provides an example for evaluating the use of iUML-g in a case study. The case study illustrates that iUML-g is more capable of modeling systems that involve many different domains.

5.1 Secured health care system (Data Warehouse +Security+GNSS)

This case study addresses the issue of system security, especially health care systems. Health care systems, placed in hospitals, handle tremendous amounts of inpatient and outpatient records. Such records store information about patients, such as personal information, financial issues, physical tests results, medical history background, and current health condition.

5.1.1 Problem description

Some hospital information is considered private and should be checked and accessed only by the concerned staff or the treating physicians. The health care system must be secure for many reasons. For example, patients' confidential and sensitive data need to be tightly locked away not only from outsiders but from non-concerned personnel, such as receptionists or laboratory staff, who are privileged to access certain information only.

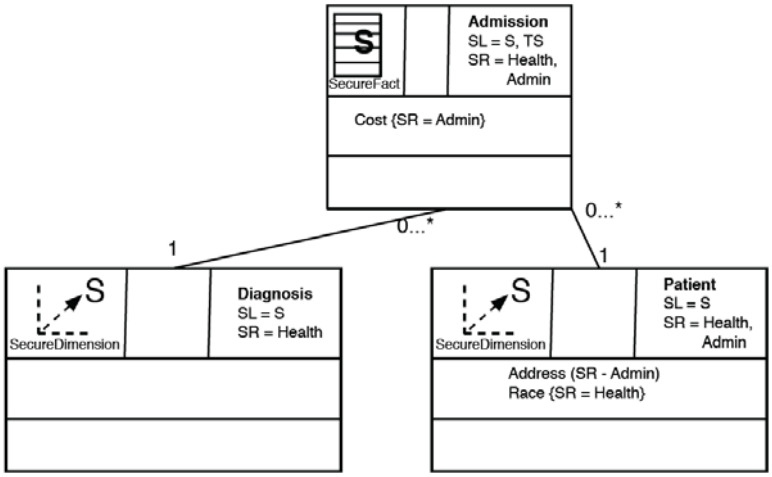


Fig. 2 iUML-g integrated classes created using Dia

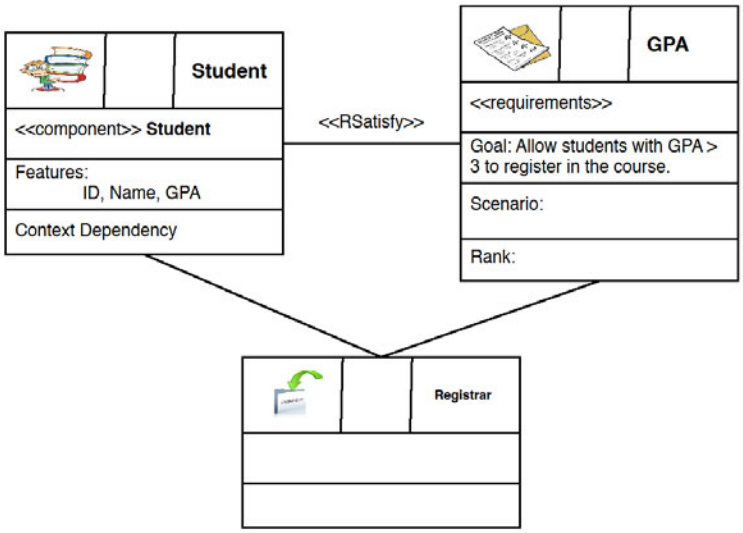


Fig. 3 iUML-g class diagram example created using Dia

Using UML to enforce security measures requires extensions to UML that add different modeling elements with different techniques, which ensure that the modeled system is secure enough. It also focuses on only one domain.

In iUML-g, the user uses one integrated form to cover security concerns for multiple domains: data warehouse and secured class diagram design. The previous extensions to UML by Peterson *et al*. (2006) and Fernández-Medina *et al*. (2007) are security techniques that are limited to specific domains. On the other hand, in iUML-g, the user can take advantage of all the integrated security techniques available to address security concerns using modeling elements, i.e., stereotypes and tagged values that are general enough to work on any problem domain.

5.1.2 Applying the iUML-g

To create the class diagram for this system, we can take advantage of the stored graphical symbols in the library. Table 10 shows the iUML-g graphical symbols that will be adopted and used in the creation of a class diagram.

The overall goal is to incorporate security packages and tiles that were previously specified into the main elements of the system, i.e., elements that need security measures, such as patients' history records, diagnosis files, and financial arrangements. These security measures will ensure that these important data are accessed only by authorized users.

First, we have to define the users of the system. Fig. 4 specifies the health and non-health employees of the hospital. This helps in defining the authorized and unauthorized users of the system.

The next step is defining the levels of security. These levels will be assigned to patients' data in their stored records. The constraints on these levels are placed on their values. The security levels must have a value range only from confidential, secret, and top secret. Fig. 5 shows the defined levels of security.

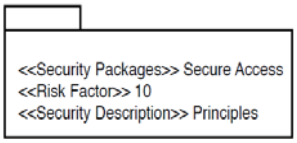
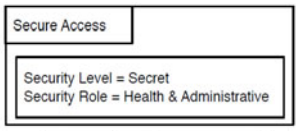
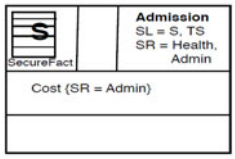
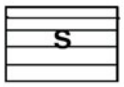
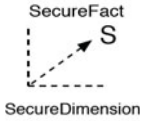
After defining the users and levels of security, we have to define the information that has to be secure. We will define the authorized users who have access to the information (security role) and what levels of security will be placed over such information (security level). Table 11 describes the different types of records that need to be secure.

Table 12 shows the assignment of security roles and levels over the hospital records. Security roles and levels are expressed as sets of tagged values.

The tagged values shown in Table 12 will now be inserted into the security tiles (Figs. 6–9).

The next step is creating security packages. Security packages have to refer to the previously defined security tiles. This is done by writing the security tile's name next to the <<Security Package>> label in the package (Figs. 10 and 11).

Table 10 Excerpt of the iUML-g library

Modeling element	Source	Meaning of the modeling element
	Peterson *et al*. (2006); Fernández-Medina *et al*. (2007)	The security package will be inserted into the class diagram and will be attached to the classes that need to be protected from security attacks. Each security package has three attributes: risk factor, which calculates the security attack; security tile, which protects the main parts of a system; and security descriptor, which describes the security categories that protect specific parts of the system
	Peterson *et al*. (2006); Fernández-Medina *et al*. (2007)	A security tile protects parts of a system. It mostly contains tagged values specified by security analysts and can be attached to security packages to cover more security concerns
	Byeon *et al*. (2004); Fernández-Medina *et al*. (2007)	A class icon with iconic representation to display graphical information along with textual information such as class name, security levels, and roles
	Fernández-Medina *et al*. (2007)	Security information and constraints
	Fernández-Medina *et al*. (2007)	Dimensions within a multidimensional model

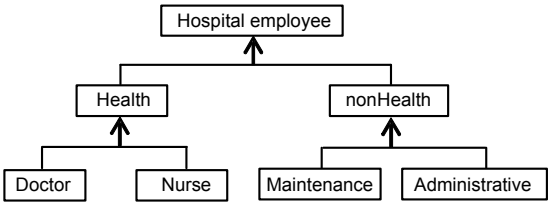


Fig. 4 Hierarchy of users as suggested by Fernández-Medina *et al*. (2007)

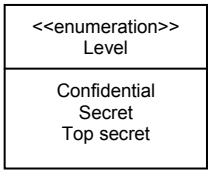


Fig. 5 Levels of security as suggested by Fernández-Medina *et al*. (2007)

Table 11 Different types of hospital records

Element	Description
Admission	Containing individual admissions of patients of one or more hospitals
Diagnosis	Containing information on each user diagnosis
Patient	Containing patients' information
Diagnosis group	Containing a set of groups of diagnosis
City	Containing information on cities
User profile	Containing the users who will access the model

Table 12 iUML-g security roles and levels

Element	Tagged value
Admission	Access by users who have Security Level = Secret & Top Secret & Security Role = Health & Administrative The attribute 'Cost' is accessed only by Security Role = Administrative
Diagnosis	Access by users who have Security Level = Secret & Security Role = Health
Patient	Access by users who have Security Level = Secret & Security Role = Health & Administrative The attribute 'Address' is accessed only by Security Role = Administrative The attribute 'Race' is accessed only by Security Role = Health
Diagnosis group	Access by users who have Security Level = Confidential
City	Access by users who have Security Level = Confidential

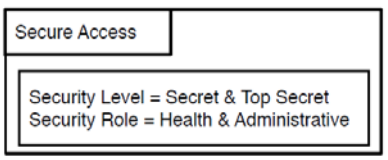


Fig. 6 iUML-g security tile #1

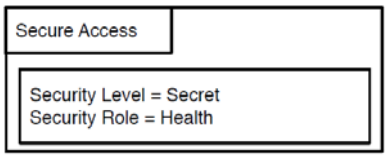


Fig. 7 iUML-g security tile #2

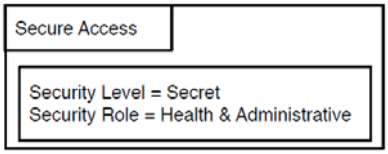


Fig. 8 iUML-g security tile #3

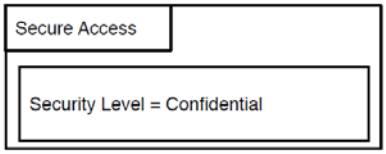


Fig. 9 iUML-g security tile #4

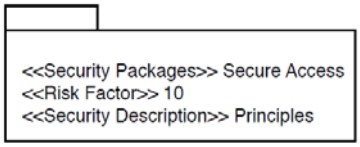


Fig. 10 iUML-g security package (secure access)

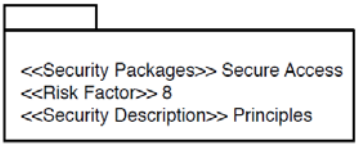


Fig. 11 iUML-g security package (secure attribute access)

The next step is to create the classes that represent the main elements of the system: Admission, Patient, Diagnosis, Diagnosis group, and City. Fig. 12 shows an example of the iUML-g class 'Admission'. The goal of this design is to have unique and helpful graphical notations attached to the created classes.

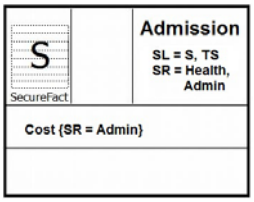


Fig. 12 iUML-g class 'Admission' created using Dia

The final step is integrating security packages into the UML class diagram (Fig. 13). Each security package protects a certain type of hospital record, which is represented as a class in the diagram.

5.1.3 Discussion

For this case study, some modeling elements were used from iUML-g to consider some issues that were not handled or addressed by UML. The graphical symbols found in this case study were used to emphasize the issue of security and how to map it graphically to the iUML-g class diagram. Fig. 10 shows an example of a security package that was especially created to be used in domains that require security measures.

Attaching graphics to classes also helps the classes to be more readable. Dividing the first row of the class vertically helps attach more information about the class in small compartments, such as iconic notations, class name, security levels, and roles. Fig. 12 shows iUML-g design of an 'Admission' class.

The essence of UML is the ability to model the targeted system using a set of graphical notations. The limited set of UML graphical notations can help the system designer to better visualize the system's internal and external elements, but at the same time, and as mentioned before, this set is limited. Unfortunately, UML was unable to address some problem domains. UML has to be adapted and extended for such domains. Fernández-Medina *et al*. (2007) applied their extension to UML for the conceptual design of a secure multidimensional model within the context of a typical health care system. Byeon *et al*. (2004) provided notational help to obtain precise measurements and precise calculations of real-world geographical entities, and Peterson *et al*. (2006) used a UML class diagram to represent an ATM model integrated with UMLpac for possible security considerations. Without extending UML, it would be challenging for UML to model a secure health care system using regular notations and other modeling elements. Stereotypes and especially tag definitions must be defined in order to enforce secure access to patients' records. Also, security packages and tiles, as discussed in this case study, create another shield to prevent such important records from security attacks. The key issue is to specify more security measures and techniques to protect the stored information.

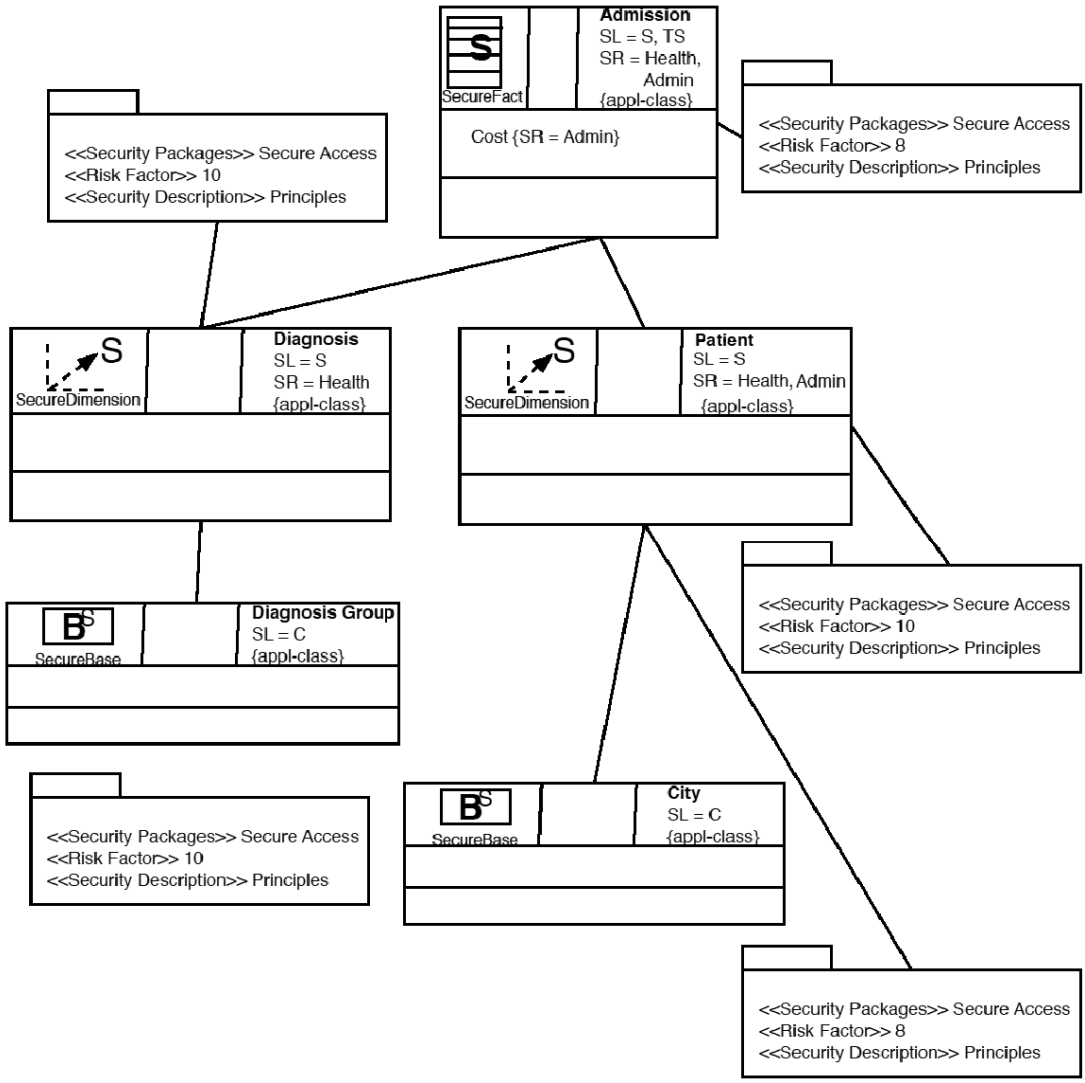


Fig. 13 Integrated UML class diagram (secured health care system)

iUML-g integrates different extensions, concerning different and similar domains, for the sake of using one comprehensive set of graphical concepts when dealing with a number of domains. Without using iUML-g, one cannot place more security techniques over the multidimensional elements such as patient, admission, and diagnosis. iUML-g handles security by setting tagged values and constraints in the data warehouse application domain, and this can be greatly enhanced, security-wise, by attaching security packages to the elements found in the data warehouse domain.

5.1.4 Threats to validity

The validity of iUML-g is threatened by two main issues: the validity of the available extensions, and the reliability of the integration process. In the former, each UML extension must provide a rich and robust extension to the UML. In this work, we assumed the validity of the available extensions, and therefore no validation of the available extensions was done from our side.

In the second threat, i.e., reliability of the integration process, the integration process must also be applied carefully. The steps of the integration process must be revised repeatedly. In this work, the proposed integration process worked well while integrating the available extensions in the literature; however, new extensions may require the process to be modified.

6 Conclusions

The rationale behind the integration process was to come up with one form of UML in order to address a variety of problem domains. In the literature, many UML extensions were proposed, each addressing a particular domain. Examples of these domains are web hypermedia applications, aspect-oriented modeling, distributed systems, component-based software systems, data warehouses, design patterns, etc., but these UML extensions are specific to particular problem domains; in other words, such extensions are not applicable to other domains. The novelty is that we provide an integrated UML that supports not just a single domain but a number of domains.

In this paper, we proposed a framework to integrate the available UML extension. We then used the framework to propose an integrated UML-graphical form. The process was verified by using a case study in which we modeled a system that uses different domains but which UML is unable to model.

Our future work will include providing an integrated UML for the extension that modifies the meta-model to provide a complete integrated UML (iUML). We also plan to consider other UML diagrams such as activity and collaboration diagrams to cover more areas in the software development systems. Other future work would include the integration of iUML-g with available integrated development environments (IDEs) such as Rational Rose or Enterprise Architect.

References

Atkinson, C., Gerbig, R., Fritzsche, M., 2015. A multi-level approach to modeling language extension in the enterprise systems domain. *Inform. Syst.*, **54**:289-307. http://dx.doi.org/10.1016/j.is.2015.01.003

Booch, G., Rumbaugh, J., Jacobson, I., 2005. The Unified Modeling Language User Guide (2nd Ed.). Addison-Wesley Professional.

Boulil, K., Bimonte, S., Pinet, F., 2015. Conceptual model for spatial data cubes: a UML profile and its automatic implementation. *Comput. Stand. Interf.*, **38**:113-132. http://dx.doi.org/10.1016/j.csi.2014.06.004

Byeon, W.S., Wang, B., Jeong, S.K., *et al*., 2004. Extension and implementation of iconic stereotype for GNSS application in the UML class diagram. Proc. Int. Conf. on Cyberworlds, p.162-169. http://dx.doi.org/10.1109/CW.2004.32

Chung, L., Supakkul, S., 2006. Representing NFRs and FRs: a goal-oriented and use case driven approach. *LNCS*, **3647**:29-41. http://dx.doi.org/10.1007/11668855_3

Cruz-Lemus, J.A., Genero, M., Caivano, D., *et al*.,, 2011. Assessing the influence of stereotypes on the comprehension of UML sequence diagrams: a family of experiments. *Inform. Softw. Technol.*, **53**(12):1391-1403. http://dx.doi.org/10.1016/j.infsof.2011.07.002

Cunha, A., Garis, A., Riesco, D., 2015. Translating between Alloy specifications and UML class diagrams annotated with OCL. *Softw. Syst. Model.*, **14**(1):5-25. http://dx.doi.org/10.1007/s10270-013-0353-5

da Silva, V., de Lucena, C.J.P., 2004. From a conceptual framework for agents and objects to a multi-agent system modeling language. *Auton. Agents Multi-agent Syst.*, **9**(1-2):145-189. http://dx.doi.org/10.1023/B:AGNT.0000019691.42633.07

Djemaa, R.B., Amous, I., Hamadou, A.B., 2006. WA-UML: towards a UML extension for modelling adaptive Web applications. Proc. 8th IEEE Int. Symp. on Web Site Evolution, p.111-117. http://dx.doi.org/10.1109/WSE.2006.20

Dong, J., 2002. UML extensions for design pattern compositions. *J. Obj. Technol.*, **1**(3):149-161.

Dong, Y., Li, M., Wang, Q., 2002. A UML extension of distributed system. Proc. Int. Conf. on Machine Learning and Cybernetics, p.476-480. http://dx.doi.org/10.1109/ICMLC.2002.1176800

Dou, L., Liu, Q., Yang, Z.Y., 2013. A metamodeling approach for pattern specification and management. *J. Zhejiang Univ.-Sci. C (Comput. & Electron.)*, **14**(10): 743-755. http://dx.doi.org/10.1631/jzus.C1300040

Fei, C., Yan, C., 2008. Spaeris: a multi-agent system specified by agent UML. Proc. Int. Seminar on Future Information Technology and Management Engineering, p.368-371. http://dx.doi.org/10.1109/FITME.2008.60

Fernández-Medina, E., Trujillo, J., Villarroel, R., *et al*., 2007. Developing secure data warehouses with a UML extension. *Inform. Syst.*, **32**(6):826-856. http://dx.doi.org/10.1016/j.is.2006.07.003

Fontoura, M., Pree, W., Rumpe, B., 2000. UML-F: a modeling language for object-oriented frameworks. *LNCS*, **1850**:63-82. http://dx.doi.org/10.1007/3-540-45102-1_4

Génova, G., Llorens, J., Fraga, A., 2014. Metamodeling generalization and other directed relationships in UML. *Inform. Softw. Technol.*, **56**(7):718-726. http://dx.doi.org/10.1016/j.infsof.2014.01.010

Hausmann, J.H., Heckel, R., Sauer, S., 2001. Towards dynamic meta modeling of UML extensions: an extensible semantics for UML sequence diagrams. Proc. IEEE Symp. on Human-Centric Computing Languages and Environments, p.80-87. http://dx.doi.org/10.1109/HCC.2001.995242

Hsia, P., Gupta, A., Kung, C., *et al*., 1995. A study on the effect of architecture on maintainability of object-oriented systems. Proc. Int. Conf. on Software Maintenance, p.4-11. http://dx.doi.org/10.1109/ICSM.1995.526522

Hsu, I.C., Ting, D.H., Hsueh, N.L., 2014. MDA-based visual modeling approach for resources link relationships using UML profile. *Comput. Stand. Interf.*, **36**(3):648-656. http://dx.doi.org/10.1016/j.csi.2013.08.017

Jantan, A.H., Sumari, P., Sulaiman, S., 2008. Com$^+$HDM: extending UML profiles for modeling complex Web hypermedia applications. Proc. Int. Conf. on Advanced Computer Theory and Engineering, p.290-294.

Lara, J.A., Lizcano, D., Martínez, M.A., *et al*., 2014. A UML profile for the conceptual modelling of structurally complex data: easing human effort in the KDD process. *Inform. Softw. Technol.*, **56**(3):335-351. http://dx.doi.org/10.1016/j.infsof.2013.11.005

Magureanu, G., Gavrilescu, M., Pescaru, D., 2013. Validation of static properties in unified modeling language models for cyber physical systems. *J. Zhejiang Univ.-Sci. C (Comput.& Electron.)*, **14**(5):332-346. http://dx.doi.org/10.1631/jzus.C1200263

Mahmood, S., Lai, R., 2013. RE-UML: a component-based system requirements analysis language. *Comput. J.*, **56**(7):901-922. http://dx.doi.org/10.1093/comjnl/bxs089

Misbhauddin, M., Alshayeb, M., 2015. Extending the UML use case metamodel with behavioral information to facilitate model analysis and interchange. *Softw. Syst. Model.*, **14**(2):813-838. http://dx.doi.org/10.1007/s10270-013-0333-9

Peterson, M.J., Bowles, J.B., Eastman, C.M., 2006. UMLpac: an approach for integrating security into UML class design. Proc. IEEE SoutheastCon, p.267-272. http://dx.doi.org/10.1109/second.2006.1629362

Saleh, K., El-Morr, C., 2004. M-UML: an extension of UML for the modeling of mobile agent-based software systems. *Inform. Softw. Technol.*, **46**(4):219-227. http://dx.doi.org/10.1016/j.infsof.2003.07.004

Sanada, Y., Adams, R., 2002. Representing design patterns and frameworks in UML—towards a comprehensive approach. *J. Obj. Technol.*, **1**(2):143-154.

Seemann, J., von Gudenberg, J.W., 1999. Extension of UML sequence diagrams for real-time systems. *LNCS*, **1618**: 240-252. http://dx.doi.org/10.1007/978-3-540-48480-6_19

Stein, D., Hanenberg, S., Unland, R., 2002. A UML-based aspect-oriented design notation for AspectJ. Proc. 1st Int. Conf. on Aspect-Oriented Software Development, p.106-112. http://dx.doi.org/10.1145/508386.508399

Xie, S., Kraemer, E., Stirewalt, R.E.K., 2007. Empirical evaluation of a UML sequence diagram with adornments to support understanding of thread interactions. Proc. 15th IEEE Int. Conf. on Program Comprehension, p.123-134. http://dx.doi.org/10.1109/ICPC.2007.19

Zhou, X.C., Liu, C., Niu, Y.T., *et al*., 2008. Towards a framework of aspect-oriented modeling with UML. Proc. Int. Symp. on Computer Science and Computational Technology, p.738-741.

Zubcoff, J., Pardillo, J., Trujillo, J., 2009. A UML profile for the conceptual modelling of data-mining with time-series in data warehouses. *Inform. Softw. Technol.*, **51**(6):977-992. http://dx.doi.org/10.1016/j.infsof.2008.09.006