Requirements Capture Workflow in Global Information Systems

M.J. Escalona, J. Torres, and M. Mejías

Department of Computer Languages and Systems, University of Seville

Abstract. The development of information systems has changed a lot in the last years. Nowadays, applications are often developed in distributed environment. It is quite common, they are distributed via Internet and they usually have hypermedia and multimedia elements in huge databases. They are characterized by having complex functional and security requirements, many and undefined users who have different degree of knowledge. These systems are named *Global Information Systems.* The development of these complex global information systems must be like a software project, based on a development methodology, to get the application suitable to the client's requirements. This methodology must offer a right treatment of all its aspects. Nowadays, there is no standard methodology which covers all these characteristics. On the one hand, there are some traditional propositions, like the Unified Process [11]. This is a good proposition to work with storage and functional requirements. On the other hand, there are propositions that have come from the multimedia environment, like OOHDM [18], Hyper-UML [14], WSDM [4], etc. which, although give more importance to the interface and navigation, don't cover all the phases of the whole life cycle. After doing a comparative study of the most relevant methodologies for hypermedia and Web development published in the last few years [7], we have made a methodology proposition to develop global information systems. This methodology is based on the Unified Process, but it adds new models and aspects to treat correctly the navigation, the hypermedia and the interface. In this paper, we present a global vision of our methodology and we focus on the proposition to get requirements from the user. To present the results, we apply the proposition to a real problem in a public company in Seville.

Keywords: Global Information System, development methodology, requirements, navigation, global interface.

.

1 Introduction

Nowadays, if a developer wants to apply a methodology to develop his global information system, he has a lot of possibilities. Global information systems are similar to classic management systems because both must deal with complex storage and functional requirements. So, the developer could use an object-oriented methodology like RUP [2]. However, global information systems have some characteristics like navigation, interface or hypermedia, which need a special treatment. In this sense, the developer could use a proposition to hypermedia applications, like OOHDM, EORM, etc. But these propositions don't cover the complete life cycle. They are often focused on design and they forget other phases like requirements capture or analysis. We have studied the actual possibilities that a developer has to apply to his global information system. In table 1, we offer an abstract of this study [7]. In this table, we can read the name of the methodology, its reference and a short description. Also, we show what life cycle phases it treats (R: Requirements capture; A: Analysis; D: Design; I: Implementation; T: Tests) and the aspects which it covers (S: Storage; F: Functional; N: Navigation; I: Interface; M: Multimedia)

In this table, we can observe that there isn't any proposition which cover the whole life cycle and deal with all global information system characteristics. However, these propositions offer models and techniques that can be applied successfully.

We have studied all these models and techniques to decide which are the most suitable. Therefore, we are developing a new methodological proposition to global systems [6]. This proposition takes some of these models and techniques and proposes others. The life cycle of our methodology is quite simple. Our methodology proposes that the project life is divided into five workflows: requirements capture, analysis, design, implementation and test. In each workflow, our methodology proposes different activities and tasks to treat correctly the classic aspects (storage, functionality, security, etc.), as well as the multimedia aspects (interface, navigation, hypermedia, etc.). The first workflow to do is the capture and the definition of requirements. In this paper, we would like to present this workflow, namely, the activities to do, the techniques that we have to apply and the structure of the workflow results.

2 Getting Requirements

In global information systems, the requirements capture is a critical workflow because the rest of the process is based on it. In this workflow, users and developers must decide what the system must offer. We have said that global information systems have some special characteristics: complex storage and functionality, navigation, interface and hypermedia. These special characteristics must be defined in the first workflow. Therefore, it's necessary that the methodology offer a mechanism to define all these aspects. However, this mechanism must be simple enough to be understood by the user as well as complete enough to be useful to the developer.

Table 1. Abstract of the actual methodology

Methodology	Description	Phases					Aspects				
		R	A	D	I	T	S	F	N	I	M
HDM [8]	It proposes a model to design hypermedia systems. It's based on ERD.			X			X		X		X
RMM [9]	It's a methodology to design hypermedia system. It is based on HDM.			X	X						X
EORM [12]	It's a methodology to design hypermedia system. It has a tool to help the developer.			X	X		X		X		
OOHDM [18][19][20]	It's a methodology to design hypermedia system. It offers new models and quite interesting techniques.			X	X		X		X	X	X
WSDM [4]	It's a methodology to design kiosk web. Its design is based on the user		X	X	X		X				X
OO-Method [17]	It's an environment to develop information systems.			X	X	X	X	X		X	
SOHDM [12][21]	It's a methodology that uses OOHDM, but that adds the scenario technique to capture requirements to it.	X		X	X		X	X	X	X	X
RNA [1]	It's a proposition to do the analysis in web information systems.	X				X	X			X	
HFPM [16]	It's a proposition that describes what process must be used to develop web systems.	X	X	X	X	X	X		X	X	
RUP [2]	It's the methodology proposed by the authors of UML	X	X	X	X	X	X	X			
Building Web Applications with UML [3]	It's a proposition which add some stereotypes to UML to design web information systems.			X					X	X	X

Our proposition is based on getting the system objectives. Starting from them, the developer has to define the information storage requirements, that is, what information the system has to store. Afterwards, actors, which are going to interact with the system, must be defined, as well as the functionality the system offers them, defining the functional requirements. When the designer knows what actors can do with the system, who can work with it and what will be stored by the system, the

interaction requirements must be identified and described. The interaction requirements capture how the information will be presented to the user and how actors will be able to use the system functionality and to do queries to the system. Finally, in this workflow, non-functional requirements, like security, communication aspects, etc., must be defined. In figure 1, a diagram of requirements capture workflow is shown.

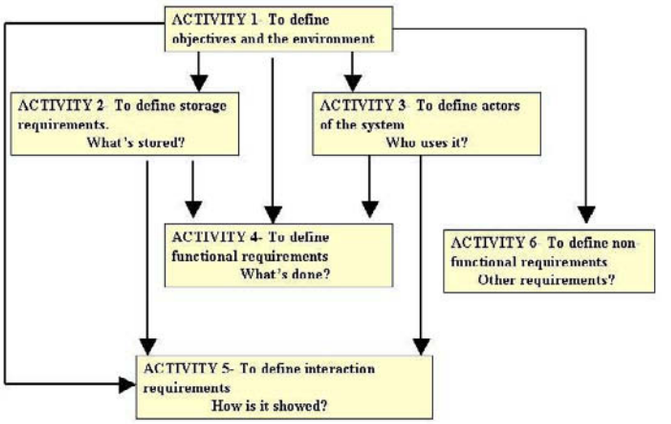

Fig. 1. Requirement capture squema

In our methodology, in order to present the result to the client, patterns have been defined. These patterns offer a structured and complete technique, but also easy to be interpreted by the client. To present the different activities, we will use a real example. A global information system to spread out and manage information about historic heritage.

2.1 Activity 1- To Define Objectives and the Environment

In this activity, the developer must:

- Study the environment and the company. To get that, it's necessary to get brochures and to study old systems. This study isn't necessary when the developer knows the company well.
- Do interviews with final users in order to get their necessities. Some interview techniques (JAD, Brainstorming, etc.) could be applied [5].
- Define the objectives. When the developer knows the necessities, he must define the objectives. In order to describe them, a pattern is used. In our example, the principal system objective is to manage the information of monuments. In table 2, this objective description is made using the pattern.

Table 2. An objective pattern to our example

OBJ-01	To manage the information of monuments
Description	The system must let manage patrimonial information. This information is divided into: Identification information, which lets identify each monument Description information, which describes each monument Graphical information, which is a set of images of each monument

An identifier names each objective. This identifier starts with OBJ, followed by a unique number. A description of the objective must be offered.

2.2 Activity 2- To Define Storage Requirements

When we know what we want to get, it's necessary to define what information is stored by the system. To describe this information, we propose to use another pattern. So, our example must store the information of monuments. The user needs the name and the address of the monument, its chronology, namely when it was built; its authors, that is who did it; its styles, such as baroque, gothic, etc.; its typology: a church, a square, etc; and a list of images where the monument appears. These are its specific data. In the pattern in table 3, we can read the description of this storage requirement. An identifier names each storage requirement, which starts with "SR", and a unique number follows it. In the 'associated objectives' row, the developer must enumerate which objectives are gotten (or are partial gotten) with this storage requirement. So, storing information about monuments we get a partial solution to manage the information of monuments.

Table 3. A storage requirement description

SR-01	Information of the monument	
Associated objectives	OBJ-01: To manage the information of monuments	
Description	The system must store information about monuments. Specifically:	
Specific data	Name and description	Nature
	Name: It's the name of the monument.	String
	Address: It's the address of the monument.	String
	Chronology: It's the date when the monument was built	Chronology
	Author: It's a set of authors or creators, who did the monument or contributed in its building.	String Cardinality: 0..n
	Style: It's the set of the monument styles.	String Cardinality: 0..n
	Typology: It's the set of the monument typologies.	String Cardinaly:0..n
	Image: It's a set of images where the monument appears.	Image Cardinality: 0..n

In the next row, a description is required. After that, the specific data are described. Each piece of specific data has a unique name, a description and a nature. The nature is the abstract type of each piece: integer, string, sound, image, etc. In our methodology, there are some predefined nature, but a mechanism to define new natures is given. In this example, there are some specific data with predefined nature: string, images, etc. But there is a new nature, namely the chronology. Each new nature must be described by another pattern, in order to clarify its meaning. In table 4, the Chronology structure is described. In this pattern, we describe the structure of the nature, its rank and its restrictions.

Table 4. A new nature description

NA-01	Chronology	
Structure	Field	Nature
	Beginning year: It's the year when the monument was started to build.	Date Format: yyyy
	Finished year: It's the year when the monument was finished.	Date Format: yyyy
Rank	Years can be positive (a. D.) or negative (b. C).	

In table 3, we can observe that some specific data can have a cardinality. When a storage requirement can have multiple values for a piece of specific data, we show the number of values in the cardinality. So, a monument can have 0 or an indeterminate number of styles or typologies.

2.3 Activity 3- To Define Actors

When the developer knows what the system has to store, he must define who can use the application. Therefore, he must define actors. An actor is a person, an external system or an external process that works with the system.

In this activity, the developer has to start defining basic actors. A basic actor is a simple role, which is identified with a specific criterion. In our system, we can classify according to two criteria. On the one hand, we can classify the actor according to his investigation area. So we have: Archaeologist (table 5) or Artistic. On the other hand, we can classify according to the actor's use in the system. So we have: Administrator or General User. An administrator is the person who can update and consult information. A general user can only consult.

In order to define basic actors, the developer has to do a pattern. We present the Archaeologist description in table 5. This actor has a unique identifier: AC-01. In the second row, objectives that are associated with this actor must be enumerated. Afterwards, the used criterion to classify this actor must be described. Finally, the developer has to describe the basic actor.

Table 5. A basic actor definition

AC-01	Archaeologist
Associated objectives	OBJ-01: To manage the information of monuments
Criterion	This is a possible actor in the system when we classify users according to his investigation area
Description	The system must offer a mechanism to work with archaeologists. An archaeologist is a person who is interested in archaeologist monuments.

After defining basic actors, we must study the incompatibility between them. Two basic actors are incompatible when a user can't use the system playing like both of them. Therefore, in our system, the same user can't be an administrator and a general user, but he could be an administrator and an Archaeologist at the same time.

In order to define actors' incompatibility, a matrix must be used. In table 6, we present our system incompatibility matrix.

Table 6. An incompatibility matrix

Basic Actor	Archaeologist	Artist	Administrator	General user
Archaeologist	-			
Artist		-		
Administrator			-	X
General user			X	-

'X' represents the incompatibility between two actors. So Administrator and General user are incompatible. Sometimes, there are other very important actors in the systems, which are complex actors. A complex actor is a role, which is composed of two or more actors (basic or complex). So, in our system we must have an artistic-archaeologist. It's an actor who is an archaeologist and an artist at the same time.

To define complex actors, we use a matrix. In table 7, we show the complex actor matrix of our system.

Table 7. Complex actor definition

	Actor				
Complex Actor	Archaeologist	Artist	Archaeologist-Artist	Administrator	General user
Archaeologist-Artist	^	^			
Archaeologist-Administrator	^			^	
Archaeologist-Artist-Administrator	^	^	^	^	
Artist-Administrator		^		^	

'^' represents that the complex actor in the row takes the same role as the actor in the column.

2.4 Activity 4- To Define Functional Requirements

When the developer knows what the system stores and who can use it, he must define the system functionality. To define functional requirements, we propose to use the standard use cases [10]. It's not necessary to explain what it is. In our example, we have a lot of use cases. One of these is the use case which describes how a user can introduce a new monument into the system. A use case is composed of a diagram and a description. In figure 2 we show a use case diagram for this example.

Table 8. A functional requirement definition

FR-01	Introducing a new monument	
Associated objectives	OBJ-01: To manage the information of monuments	
Description	The system has to do this use case when the user wants to introduce a new monument into the database	
Precondition	The user has to be an available user in the system	
Actors	AC-04: Administrator	
Normal sequence	Step	Action
	1	Execute the option "New".
	2	The system accepts the user.
	3	The user introduces the address of the monument.
	4	The system asks for the name of the monument.
	5	The user introduces the name.
	6	The system introduces the new monument in the database and the user is allowed to introduce the information of the monument (images, chronology, etc.).
	7	The user introduces the rest of the information.
	8	The systems goes back to the main menu.
Postcondition	Nothing	
Exceptions	Step	Action
	1	The user is not an available user in the system, so the system doesn't let introduce a new monument
	5	The monument is already in the system. So an error message is showed
Approximate frequency	10 times a month	

Each use case has to be described using a pattern. In table 8, we describe use case diagram in figure 2. In this pattern we must identify the use case with a unique identifier (FR-01). We must justify why this use case is defined, therefore it's necessary indicate associated objectives. Afterwards, it's necessary to do a description and indicate the functional requirement precondition. In the 'Actors' row, the developer has to enumerate actors who can execute this use case. To complete the use case description, normal execution sequence, postcondition and exceptions of this use

case must be described. Eventually, if it's possible, the developer has to give an approximate frequency.

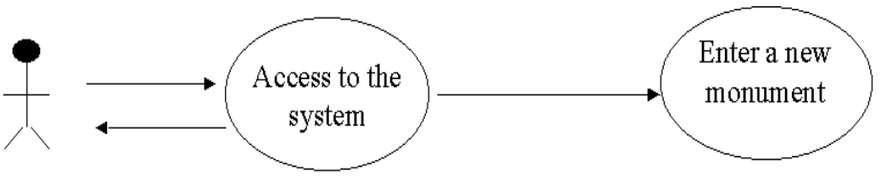

Fig. 2. A use case diagram

2.5 Activity 5- To Define Interaction Requirements

At this point, the developer knows what the system has to store, who can use it and what can be done with it, but it isn't enough. In global information systems, interface and navigation are critical aspects, so we have to define how the information will be presented to the user. In order to define this, the developer has to define interaction requirements using patterns again. Each interaction requirement is a node, in which specific information is showed. Information in a node is about a specific subject. Therefore, in our system we can define a node, which shows information about the identification of monuments. In this node, we must show information, which identifies the monument: its name, its address and its images.

A unique identifier names each interaction requirement. In the pattern of table 9 is IR-01. For each interaction requirement, the developer has to enumerate the actors who can interactuate with it. Moreover, a description must be defined. If some entry parameters are necessary in order to show this requirement, they will be described in the fourth row. In the 'associated functionality' row, the developer must enumerate functional requirements, which can be executed in this interaction requirement. Afterwards, the developer has to mention specific data of storage requirements in order to detail the information showed by the node. An interaction requirement can show specific data of different storage requirements and a piece of specific data can be showed in different interaction requirements. So, in this row, we must indicate the storage requirement identifier and the names of each piece of specific data.

Eventually, we must indicate navigation. From this information requirement, the user will be able to navigate to other information requirements, so the developer must enumerate them in the 'Exit' row. As the same time, from other information requirements, the user can navigate to this information requirement. Those are enumerated in the 'Entry' row. In our example, the user can get IR-01 information requirement from IR-03 and can navigate from IR-01 to IR-02 too.

2.6 Activity 6- To Define Non-functional Requirements

To conclude, it could be possible that there are other requirements which aren't storage, functional or interaction requirements. These are non-functional requirements. In the last workflow activity, the developer has to describe other requirements like security, efficiency, etc. To describe these requirements, we propose another pattern.

For instance, in our system we can indicate that it must offer a mechanism to do automatically backups. In table 10, we describe this requirement.

Table 9. An interaction requirement definition

IR-01	Monument identification information
Actors	AC-01: Archaeologist AC-02: Artist
Description	The system has to show information about the identification of monuments.
Entry parameters	The user must introduce the name of the monument.
Associated functionality	FR-01: Introducing a new monument
Showed information	SR-01.Name SR-01.Address SR-01.Image
Exit	IR-02
Entry	IR-03

Table 10. A non-functional requirement definition

NFR-01	Backups
Associated objectives	OBJ-05: Get a safe system to manage monuments
Description	The system must offer an automatic mechanism to do backups and to recover the information when it is necessary.
Commentary	This is a very important requirement because in this system there is very important information

Again, we must identify each non-functional requirement with a unique identifier: NFR-01, in this case. Moreover, we must indicate its associated objectives and give a short description. In addition, we can indicate other commentaries about the requirement.

3 Requirements Capture Document

To present results of the requirements capture workflow, the developer must do the requirements capture document. Our methodology proposes a structure to this document.

The first page of the requirements capture document must be a cover. The cover must include the version, the name of the document and the date. If the organization has a predefined cover, it can be assumed. Afterwards, an index, a figure index and a table index must be include.

Table 11. Searching matrix structure

	OBJ-01	OBJ-02	…	OBJ-0n
IR-01	X	X		
IR-02				X
...				
AC-01	X			X
AC-02		X		
...				
FR-01	X	X		X
FR-02		X		
...				
IR-01	X	X		
IR-02				X
...				
NFR-01	X			
NFR-02		X		
...				

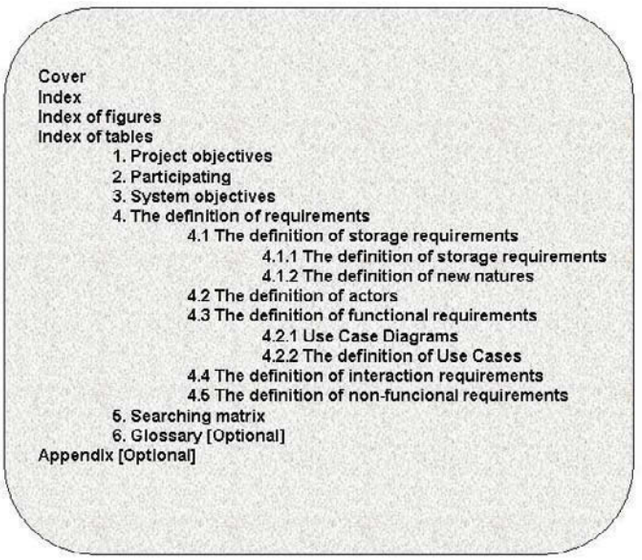

Fig. 3. The structure of the Requirements capture document

After these pages, the developer must include definitions of objectives and requirements. Therefore, the developer has to write patterns of objectives, defined in activity 1, patterns of storage requirements and new natures (activity 2). After these patterns, patterns of actors and functional requirements (defined in activity 3 and 4) must be included.

Afterwards, the developer has to describe interaction and non-functional requirements, using patterns defined in activity 5 and 6. To finalize, the developer must design a matrix. This matrix shows what objectives are associated to a specific requirement. In table 11, we show the structure of this matrix.

To finalize, in figure 3 we show the structure of the requirement documents following our proposition.

4 Conclusions

In this paper, we have presented our proposition to do the requirements capture applying our idea to a real information global system in a public company.

This proposition has been applied to several real global information systems, giving very good results. Using patterns to define system requirements and objectives is a very interesting idea. Users who are not expert in computer science, understand patterns easily. But patterns present complete and non-ambiguous information to the developer, who can use them in analysis and design.

Nowadays, we are developing a proposition to do the analysis workflow in global information systems. This analysis proposition is based on the requirements capture presented in this paper. The use of patterns is a good technique to get the analysis class diagram and interface prototypes cuasi-automatically.

Moreover, we are developing a tool which lets generate the patterns easily. Patterns are structured, so it's quite easy to introduce their information in a database, which produce the requirements capture document automatically.

References

1. M.Bieber, R.Galnares And Q. Lu. *Web engineering and flexible hypermedia.* The 2nd Workshop on Adaptative Hypertext and Hypermedia, Hypertext 1998.
2. G. Booch, J. Rumbaugh, I. Jacobson. *Unified Modeling Language User Guide*. Ed. Addison-Wesley, 1999.
3. J. Conallen. *UML Extension for Web Applications 0.91*. Available in http://www.conallen.com/technologyCorner/webextension/WebExtension091.htm .
4. O.M.F. De Troyer, C.J. Leune. *WSDM: A User Centered Design Method for Web Sites*. Tilburg University, Infolab. 1997
5. Durán. *Un Entorno MetodolÛgico de IngenierÌa de Requisitos para Sistemas de InformaciÛn.* Department of Language and Computer Science. University of Seville. September 2000. Available in http://www.lsi.us.es/~amador .
6. M.J. Escalona, J.Torres, M.Mejías. *Propuesta de metodologÌa para el desarrollo de sistemas para el tratamiento de bibliotecas digitales.* Internal Report, 2-2000. Department of Language and Computer Science. University of Seville. Seville, June 2000. Available in http://www.lsi.us.es/~informes .

7. M.J. Escalona. *MetodologÌa para el desarrollo de sistemas de informaciÛn global: an· lisis comparativo y propuesta*. Department of Language and Computer Science. University of Seville. Seville, January 2002. Available in http://www.lsi.us.es/~informes .
8. F. Garzoto, D.Schwabe and P.Paolini *HDM-A Model Based Approach to Hypermedia Application Design*. ACM Transactions on Information System, 11 (1), Jan 1993, pp 1-26.
9. T.Izakowitz, E.Stohr, P. Balasubramaniam: *RMM:A methodology for structured hypermedia design*. Comm. Of ACM, October 1995, pp.34-35.
10. Jacobson. *Modeling with use cases-Formalizing use-case modelling*. Journal of Object-Oriented Programming, June 1995.
11. Jacobson, G. Booch, J. Rumbaugh. *The Unified Software Development Process*. Ed. Addison-Wesley, 1999.
12. D.B. Lange. *An Object-Oriented Design Approach for Developing Hipermedia Information Systems.* Research Report RT00112, IBM Research, Tokyo Research Laboratory, Japan, 1995.
13. H. Lee, C. Lee & C. Yoo. *A Scenario-based object-oriented methodology for developing hypermedia information systems*. Processing of 31st Annual Conference on Systems Science. Eds. Sprague R.
14. L. Mandel, A.Helmerich, L.A. Olsina, G.Rossi, M.Wirsing, N.Koch. Hyper-UML. Specification and modeling of multimedia an Hypermedia Applications in Distributed systems. August 2000.
15. J. Nanard, M. Nanard. *Hypertext design environments and the hypertext design process*. Communication of the ACM, August 1995. Vol 38(8), 49-56. 1995.
16. L. Olsina. Building a Web-based information system applying the hypermedia flexible process modeling strategy. 1st International workshop on Hypermedia Development, Hypertext 1998.
17. O. Pastor, E.Insfran, V. Pelechano, J.Romero and J. Merseguer. OO-METHOD: *An OO Software Production Environment Combining Conventional and Forma Methods*. CAiSE'97. International Conference on Advanced Information Systems, 1997.
18. G. Rossi. *An Object Oriented Method for Designing Hipermedia Applications.* PHD Thesis, Departamento de Informática, PUC-Rio, Brazil, 1996.
19. Schwabe, D., Rossi, G.. *An Object Oriented Approach to Web-Based Applications Design*. TAPOS – Theory and Practice of Object Systemss, vol. 4, 1998.
20. Schwabe, D. Rossi, G. *A Conference Review System with OOHDM*. 1st International Workshop on Web-Oriented Software Technology. Valencia, Junio 2001.
21. W. Suh, and H. Lee, *A Methodology for Building Content-oriented hypermedia systems* The Journal of Systems and Software, Vol. 56, 2001, pp. 115-131.