NDT: a methodology to deal with the navigation aspect at the requirements phase

M.J. Escalona, A.M. Reina, J.Torres, M.Mejías
Department of Computer Languages and Systems. University of Seville
mjescalona@us.es

Abstract

The proliferation of the internet has provoked the conception of new design methodologies to deal with the new concepts that have appeared in web applications and weren't treated in traditional methodologies. One of these concerns is navigation. It has been proved that traditional abstraction mechanisms (functions and objects) do not treat very well some concerns that are scattered all over the program code. In this paper, we think about navigation as an aspect that can be treated independently during the development of the system. We introduce a process to identify and define such an aspect during the requirements specification phase. This process is part of the NDT (Navigational Development Technique) development process, a new approach to develop web information systems

1. Introduction

The separation of concerns is one of the main principles of software engineering. The success of the internet and multimedia has made some new design methodologies arise. These new methodologies apply this principle and address the new concepts that were not treated in traditional applications.

One of the terms that have come up in this new environment is navigation. Navigation includes the intrinsic notion of movement through an information space. The current position or context is a crucial factor, because many times the next step to give while navigating will depend on the previous navigation.

If we analyze the trends of the last web design methodologies (OOHDM [21], UWE[16], WebML[6], OOH[7], etc) and some comparatives studies [3][10][13][15], it turns out that navigation is very important and it should be taken apart from the basic functionality of the application.

On the other hand, the term aspect has been coined in the field of the advanced separation of concerns as an abstraction to represent those concepts that are scattered all over the program code.

However, according to [2], before encapsuling crosscutting behaviour into an aspect, the developer must first identity it in the requirements. This paper introduces a process that guides us to face up to the navigational aspect starting at the requirement specification stage. The description of the navigation in the requirements phase is based on some techniques that give a structured definition of it. Besides, the customer will understand it easily, and will be allowed to participate in the definition of requirements process. Thus, this will help us to evaluate if the definition of the system fits well into the requirements from the very beginning of the life cycle.

This requirements treatment proposal is the NDT [11][12] requirements phase. NDT is a new proposal to develop web information system. NDT covers two phases in the development process: requirements elicitation and analysis. In this paper, we show the first one and how it can be use to deal with the navigation aspect in web systems.

In section 2, we give a short description of the navigation concept, as it is understood in the scope of this paper. Besides, we analyse why is interesting to deal with the navigation as an early aspect. Section 3 introduces an example based on a real system to apply our proposal. Later, we will discuss the NDT process applying to the example in section 4. Finally, we will conclude the paper.

2. The development process in web methodologies

Navigation is a key concept used in web environments. Its original meaning was bound to the hypertext idea, and it defined the action of jumping from page to page through a hyperlink. The advance of technology has made us have a wider vision of the navigation concept, being not only the action of visiting page after page, but the idea of moving through an information space.

This extension of the semantics of the navigation concept invite us not to treat as navigation the fact of pushing the link, more when we browse the results of a

query processed by a search machine as *google* or *AltaVista,* since we wouldn't move from an information space to another one. This problem is considered as an interface one (we have an information space and we can't display all the information to the user simultaneously). Although we have a link in the hypertext, this link should be considered just as a way to do scrolling.

The context is a very important concept in navigation. Let us suppose that we have a web-based application for a museum and we want to obtain some information on a concrete painting. If we reach the information navigating through the author, and then we push the link Next, we will move to the next painting by the same author. However, if we access to the painting through a pictorial movement, the result of navigation pressing on Next will be different. In this case, we should move to another painting related to the same movement.

Navigation is defined as the sensation the user has when navigates through an object space from the application domain [23], but distinguishing these objects from conceptual objects, as they are customized according to the user's profile and the task is being made.

Classically, in the bibliography some groups have proposed to deal with the navigation as an aspect, but basically, they have worked in the design or implementation phases[14]. Nowadays, however, some research groups are detecting that to raise the treatment of aspects on the first phases of the life cycle gives some advantages like a early detection of mistakes and unconsciousness and the early detection of the critic interrelation between aspects that are necessary of being defined to obtain the architecture of the systems[1][14][17][18].

In order to get these advantages, we have proposed NDT (Navigational Development Technique) as a technique to deal with the aspect of the navigation in the first phases of the life cycle.

3. A practical real example

To explain the definition of the requirements technique that it is going to be introduced in the paper we will use a real example. This example let us highlight how powerful the technique is. We have chosen a system to spread out and manage the Andalusian patrimony on the Internet. This system is fairly complex, not only for its storage needs (among of them are multimedia elements), but for the functional complexity and the variety of actors that has. As the whole example is too complex, we are going to work with part of it. Let us suppose that the system stores the following data for a monument: its name, its address, a description, if its access is public or private, its chronology and a list of images. The system should store the list of historic periods, the artistic styles and the typologies of the goods, and the authors associated to it, too.

On the other hand, two kinds of users (artists and archaeologists) should access to the system. In the real system, both of them are very different, but in our example the differences are smoother. Artists are interested in the authors, and archaeologists aren't and, at the same time, archaeologists are interested in historical periods, and artists aren't. Furthermore, a user can log in using the two profiles at the same time, in which case he will browse all data.

Other classification consists of dividing them into cataloguers and consulters. The first ones are allowed to enter, modify and query the data, whereas the second ones only are allowed to query them.

The information about goods is going to be managed using two modules, one for identification data, which will contain the name, the address and the access; and the other one for description data, where description, chronology, images, styles, periods, typologies and authors will be included.

4. The process of NDT to deal with requirements

The requirements definition process that we introduce has a few interesting characteristics. First of all, it is pattern-oriented. That is, the definition and capture of requirements is made using a set of patterns or templates [9][20]. These patterns let us define in a structured way the system requirements and objectives. The structure defined by patterns, made the definition of objectives be clear and easy to understand by customers. Thus, they can participate in the process improving the detection of errors because this process can be made during the first stages of the life cycle. But, in addition, patterns offer a structured definition that will be very useful during the next stages in the life cycle [11].

The requirements definition process starts with the definition of objectives and follows with the identification of the storage needs of the system. The next step is to define and identify the roles of the system's actors and then it is followed for the definition of the functional needs and interaction requirements. The last of them are the most interesting because they let us define the navigation of the system. At last, the process ends defining other requirements, the non-functional ones, which involves other

requirements that weren't treated before. Figure 1 depicts an activity diagram of all the process. As can be observed, the requirements phase produces as result the requirements catalogue. That is the document which describes all the requirements of the system. The structure of this catalogue is defined by NDT and it is the base for the users' requirements validation. Also, it includes all the requirements description in order to be the start point in the analysis phase.

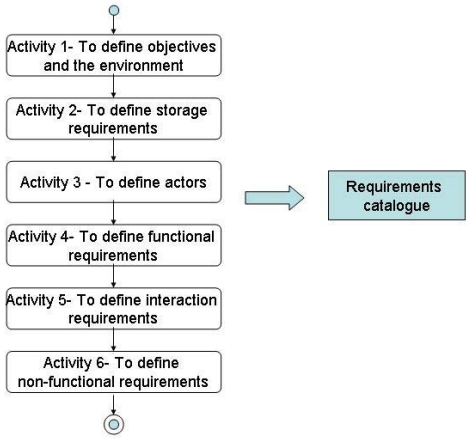

Figure 1. Scheme of the definition requirements process.

In the following sections we are going to introduce these activities briefly, focusing our attention in the fifth one. We aren't going to discuss the last one because it is out of the scope of this paper.

4.1 Identification and definition of objectives

The system objectives must answer the question: What should be gotten by the system? To do it, we propose the use of a pattern. Table 1 shows the definition of one of the objectives of our system.

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

Table 1. An objective pattern to our example

Every objective is going to be identified by a code beginning with OBJ, and is going to be followed by a number. Moreover, it will come with a brief description.

4.2 Identification and definition of the information storage requirements

The information storage requirements must answer the question: What should be stored by the system? To explain how identify and define these requirements, we are going to introduce the pattern we have used for it. Table 2 shows the pattern that indicates what should be stored about every monument by the system.

SR-01	Information of the monument	
Associated objectives	OBJ-01: To manage the information of monuments	
Description	The system must store information about monuments.	
Specific data	**Name and description**	**Nature**
	Name: It's the monument name	String
	Address: It's the monument address	String
	Description: Short description	String
	Access: It indicates if the monument can be visited or not	Numerated Values: {public, private}
	Chronology: It's the date when the monument was built	Chronology
	Author: It's a set of authors who did or contributed in the monument building.	String Cardinality: 0..n
	Historic Period: It's a set of historic period of the monument	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 Cardinality: 0..n
	Image: It's a set of images where the monument appears.	Image Cardinality: 0..n

Table 2. A storage requirements description

Every information storage requirement is identified by a code beginning with SR and it is followed by a sequential number. After that, the list of objectives fulfilled totally or partially with the definition of this requirement is enumerated. Then a brief description about the meaning of the requirement should be given and the specific data should be described. The specific data involves the data that are going to be stores about the defined requirement. Thus, for every monument we should store its name, its address, its description, its access, and so on. Every specific data is identified by a name, a description and a nature. The nature expresses the specific type of data in an abstract way without giving design details. In our process there are some predefined natures, as string, image or enumerated. But some new natures that aren't included in the predefined ones can be specified. This fact let us offer more semantic power to the definition of the requirements. To define the natures we propose a new pattern. Table 3 depicts the definition of the nature Chronology that has appeared in the requirement SR-01.

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).	
Restrictions	Beginning year must be lower than finished year.	

Table 3. A new nature description

We can see that it is identified by a code and a name, and after that, its structure, its range of values and the set of constraints that should be fulfilled are defined.

3. Identification and definition of actors

Web information systems can vary widely depending on the user who interacts with it. Systems should offer an appropriate interface and navigation depending on the role the user is playing in the system. The process we propose for the definition of the navigation needs is aware of these aspects, therefore a classification of the actors is made having in mind to define the other requirements based on the role every user can have in the system.

The definition of actors begins making a classification of the actors who can interact with the system. A few different classification criteria can be studied. The set of actors resulting is known as *basic actors*.

In our example there are four basic actors: artists, archaeologists, cataloguers and consulters. The first and the second ones have arisen due to the classification of the actors according to their research profile, whereas the third and the fourth ones take place when we classify them according to their interaction with the system.

Every basic actor has to be defined using a pattern. Table 4 shows the definition of the archaeologist pattern.

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.

Table 4. A basic actor definition

Again we have to identify every actor with a code, in this case AC-01. Then we indicate the associated objectives, the classification criteria and a brief description of it.

After defining the basic actors, it is necessary to study the incompatibilities among them. Two basic actors are incompatible if one actor can't play both roles at the same time. To define these incompatibilities we use a two-dimensional array, where the actors are placed per row and per column. An 'X' means the actors of the row and the column are incompatible. A hyphen ('-') means it isn't necessary to study the incompatibility of a role with itself. Table 5 depicts the array for our system. We can see the consulter role and the cataloguer role are incompatible, whereas a user could be archaeologist and cataloguer at the same time.

Basic Actor	Archaeologist	Artist	Cataloguer	Consulter
Archaeologist	-			
Artist		-		
Cataloguer			-	X
Consulter			X	-

Table 5. An incompatibility array

In many systems, there is what we have called *derived actors*. When one actor plays a few compatible roles simultaneously, he can play a derived role which needs a special treatment in the system. This kind of role is what we have defined as a derived actor. In our little example there isn't any derived role, but our technique offers a system to identify and define this kind of case.

4.4 Identify and define the functional requirements.

Following the recommendations of many methodological techniques [22][16][6][7], we propose the use cases to define and capture the functional requirements [4]. This technique gives us a way to represent the functional needs of the system easily and intuitively and nowadays it is widely extended. The functional requirements are defined by the use case diagrams and a textual, pattern-based representation. Every functional requirement will be identified by an identifier which begins with FR and will be followed by a unique sequential number. This identifier will be useful for the interaction requirements to reference the basic functionality associated to every actor while navigating.

4.5 Identify and define the interaction requirements.

The study of all the requirements we have enumerated is by means of defining what we know as *interaction requirements*. These requirements allow us to define the navigation through the system, indicating what has to be displayed to the user every time, depending on the role he play in that moment and the place where he is.

The requirements definition is given using two different aspects, the recovery criteria definition (or phrases), and the browsing prototypes definition.

The first of them indicates which search criteria are wanted by the user. A critical point is how these search criteria are going to be defined during the requirements specification. If we use a languages based on logical and mathematical expressions, an unskilled user won't understand it. Because of that, it is necessary to look for other ways to express this information recovery in such way it is understood by the user, and at the same time, it is adequate to the development group. In our proposal, the information recovery needs are represented with *phrases*. These phrases should be written using the BNL (Bounded Natural Language) [5]. It is a language based on defining the search criteria the user wants using the natural language. This made the results obtained be easily understood by the customer. However, if the language representing the recovery needs was totally opened, the natural language variability would make the obtained results be so ambiguous that they weren't useful. To avoid these problems, we propose an intermediate solution where the information recovery criteria are described using a set of phrases. These phrases consist of three elements:

- Fixed structures L-patterns that can't be modified
- Gaps to be filled in by the customers during the queries
- Concepts to determine what is going to be queried

The phrases introduce specified recovery criteria on the specific data from the storage requirements defined in section 4.2. For each defined nature, our technique offers a set of possible phrases. So, let us suppose our user wants to recover information on the monuments depending on its styles. The definition of the phrase will be done following a pattern similar to the one depicted in Table 6.

PH-01	Asking for authors	
Associated objectives	OBJ-01: To manage the information of monuments	
Description	Body	Actors
	SR-01.style must be exactly _____	AC-01: Archaeologist AC-02: Artist
	SR-01.style must contain this string _____	AC-02: Artist

Table 6. A phrase definition

If we look at this table, we can realize that archaeologists and artists want to do recoveries indicating the style value exactly and only artists want do recoveries with the contents.

The other elements to specify the interaction are the browsing prototypes. A browsing prototype shows how the information will be browse by the user, how navigate and how the functionality can be executed. To do it, it essential to have in mind all the elements we have defined, because the browsing prototypes will reference them.

For example, in section 2 we said the information is going to be shown to the user using modules, the identification one and the description one. In Table 7 we can realise how the browsing prototype of the identification module is defined. We can see the prototype name and its code beginning with IR and followed by a sequential number. Then we have its associated objectives and the actors who can browse it. The actors are those referenced by the definitions given during the actor's identification and definition activity. Afterwards, a description of the prototypes is given. Later the phrases executed before going into the prototype are mentioned. Moreover, the functionality associated to the prototype is written, referencing to the use cases defined during the identification and definition of the use case activity. This functionality is conditioned by the actor who is interacting with the system in that moment.

IR-01	Monument identification information
Associated objectives	OBJ-01: To manage the information of monuments
Actors	AC-01: Archaeologist AC-02: Artist AC-03: Cataloguer AC-04: Consulter
Description	The system has to show information about the identification of monuments.
Phrases	PH-02: Asking for name PH-03: Asking form access
Associated functionality	FR-01: Consulting monuments If actor = AC-03 FR-02: Introducing a new monument
Showed information	SR-01.Name SR-01.Address SR-01.Access
Exit	IR-02
Entry	Any

Table 7. An interaction requirement definition

Then, the information browsed in the prototype is shown and the specific data of the storage requirements are referenced. At last, the prototypes to where the user can navigate are shown (IR-02), and those from which the user can access to the actual prototype (in this case, there isn't any of them).

IR-02	Monument description information
Associated objectives	OBJ-01: To manage the information of monuments
Actors	AC-01: Archaeologist AC-02: Artist AC-03: Cataloguer AC-04: Consulter
Description	The system has to show information about the description of monuments.
Phrases	PH-01: Asking for styles

Associated functionality	FR-01: Consulting monuments If actor = AC-04 FR-02: Introducing a new monument
Showed information	SR-01.Name SR-01.Description SR-01.Chronology SR-01.Image If actor = AC-01 SR-01.Historic Period SR-01.Style SR-01.Tipology If actor = AC-02 SR-01.Author
Entry	IR-01

Table 8: An interaction requirement definition

Table 8 depicts the prototype for the description of the system data. Its structure is similar to the one shown above, but we must remember that there is data in which only are interested archaeologists (the period), and another ones in which only are interested artists (the authors). Table 8 shows how the pattern conditions the browsed information depending on the actor we are working with.

The set of browsing prototypes allow us to indicate what, how and to whom the information is shown, how we can navigate and what it can be done.

5. Conclusions

If we look at the trends that are following the web-design methodologies, we will realize navigation is a key concept in this kind of applications. The most recent methodologies treat it separately, in such a way that they spend a whole stage during the design phase to specify it and, also, they propose separate models.

This separation should be applied from the very beginning during the first stages of the life cycle, because navigation is so important that it requires the user participation during its definition. In this paper we have showed a technique to identify and define the requirements of a web information system, and specially the interaction requirements, because they let us define navigation easily and understandably. But, at the same time, the proposed technique is complete and rich enough to be understood by the development team. The definition of these prototypes can be used during the following development phases: analysis, design and implementation in order to define navigation according to the customer specifications.

This approach, NDT, has been applied in several real projects with companies [12]. We have got very good results in this application because patterns are very easy to be understood by users and, also, they are very useful for the development team. However, sometimes, in big projects, it is very difficult to manage them. For this reason, we have developed a tool, named NDT-Tool [12] which helps to apply all the techniques proposed by NDT.

As a future work, we want to continue working with companies in real projects, which helps us to get better results in our approach. Also, we want to continue developing NDT getting more advantages from patterns. Nowadays, we are working in the next phases: design and implementation, and we are trying to find easy ways to get the design and the implementation results from the requirements and analysis models.

6. References

1. Araújo, J., Moreira, A., Brito, I., Rashid, *A. Aspect-oriented requirements with UML*. Second International Workshop on Aspect-Oriented Modeling with UML. September 2002
2. Baniassad, E., Clarke, S. *Theme: An Approach for Aspect-Oriented Analysis and Design*. 26th International Conference on Software Engineering ICSE 04. pp 158-167. Edinburgh, Scotland, 2004.
3. Barry, C. & Lang, M. A Survey of Multimedia and Web Development Techniques and Methodology Usage. IEEE Multimedia. 2001, 52-60.2001
4. Booch G., Rumbaugh, J., Jacobson, I. Unified Modelling Language User Guide. Addison-Wesley.2000
5. Brisaboa, N.R., Durán, M.J.,Lalín, C., López, J.R., Paramá, J.R., Penabad, R., Places, A.S. A*rquitectura para Bibliotecas Virtuales*, 1999
6. Ceri, S. Fraternali, P., Bongio, A., Brambilla M., Comai S., Matera M. *Designing Data-Intensive Web Applications.*Morgan Kaufman. 2003
7. Cachero, C. OO-H: una extensión a los métodos OO para el modelado y generación automática de interfaces hipermediales. P.h. Tesis. Univ. de Alicante, 2003.
8. Cunliffe, D., Taylor, C., Tudhope, D.. *Query-based Navigation in Semantically Indexed Hypermedia.* Pr. Hypertext'97 Conference. 1997
9. Durán A. *Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información*. Tesis Doctoral. Departamento de Lenguajes y Sistemas Informáticos. Uni. de Sevilla. 2000.
10. Escalona, M.J., Mejías, M., Torres, J. *Methodologies to develop Web Information Systems and Comparative Analysis***.** Informatik/Informatique. núm. 2/2002 de I/I.
11. Escalona, M.J., Mejías, M., Torres, J., Reina, A.M. *The NDT Development Process*. The International Conference on Web Engineering.

(ICWE 2003). Springer Verlag. LNCS 2722. Pág.463-467. ISSN: 0302-9743. Oviedo, España.
12. Escalona, M.J., Mejías, M., Torres, J.. *Developing systems with NDT & NDT-Tool*. 13th International Conference on Information Systems Development: Methods and Tools, Theory and Practice. Vilna, Lituania. September 2004
13. Escalona, MJ., Koch, N. *Requeriments Engineering for Web Applications: A Comparative Study*. Journal on Web Engineering, Rinton Press, 2004, Vol2. Nº3. pp 193-212.
14. Katera, M., Katz, S. *Architectural views of aspect*. International Conference on Aspect-oriented software development. Pp. 1-10. 2003
15. Koch, N. *A Comparative Study of Methods for Hypermedia Development.* Technical Report 9905. Ludwig-Maximilian-University, Munich, Germany. 1999
16. Koch, N. *Software Engineering for Adaptive Hypermedia Applications*. Ph. Thesis, FAST Reihe Softwaretechnik Vol(12), Germany 2001
17. Rashid, A., Sawyer, P., Moreira, A., Araújo, J. *Early aspects: a model for aspect-oriented requirements engineering*. Proc. IEEE joint Conference on Requirements Engineering. Pp. 199-202. September 2002.
18. Rashid, A., Moreira, A., Araújo, J. *Modularisation and composition of aspectual requirements*. International Conference on Aspect-oriented software development. pp. 11-20. 2003
19. Reina, A.M. *Visión General de la Programación Orientada a Aspectos.* Artítulo técnico, LSI-2000-11 del Dpto. Lenguajes y Sistemas Informáticos. Univ.Sevilla. Diciembre, 2000.
20. J. Robertson and S. Robertson. *VOLERE. Requirements Specification Templates*. 2000
21. G. Rossi. *An Object Oriented Method for Designing Hipermedia Applications.* PHD Thesis, Dep. de Informática, PUC-Rio, Brazil, 1996.
22. D.Schwabe, G.Rossi. *The Object-Oriented Hipermedia Design Model*. Communications of the ACM, 38(8), August, 1995, pp.45-46
23. Tarr, P., Ossher, H., Harrison, W., Sutton, S.M. *N degrees of separation: Multi-dimensional separation of concerns.* Proceedings of the 21st International Conference on Software Engineering May 1999.