# Conceptualization of Navigational Maps for Web Applications

Antonio Navarro, José Luis Sierra, Alfredo Fernández-Valmayor, Baltasar Fernández-Manjón
*Dpto. Sistemas Informáticos y Programación*
*Universidad Complutense de Madrid*
*{anavarro, jlsierra, alfredo, balta}@sip.ucm.es*

## Abstract

*The characterization of navigational maps for Web applications helps its developers in the process of construction of these applications and it also helps users while performing their browsing. This paper presents the Pipe approach for characterizing navigational maps for Web applications at their conceptualization stage. Although Pipe was originally intended as a hypermedia-oriented notation it has been successfully used for characterizing navigational maps for Web applications. In addition, navigational maps described in terms of the Pipe notation can be easily mapped to UML-Conallen class diagrams at the design stage. This paper presents the characterization of the travel agency's navigational map using Pipe notation at the conceptualization stage. The paper also demonstrates the mapping of this Pipe navigational map to UML-Conallen class diagrams.*

## 1. Introduction

The development of Web applications is not an easy task. In the development process of this type of applications, at least three model types must be provided: *conceptual*, *navigation* and *presentation* [15]. Conceptual/structural model describes the organization of the information managed by the application in terms of the contents that constitute its information base and semantic relationships that exist among them. Navigation model describes the facilities for accessing information and for moving across the application content. Presentational model describes the way in which application content and navigation commands are presented to the user [18].

Regarding navigation model, *navigational maps* can be a very useful tool. These maps describe a global view of a Web application for an audience [22]. At present, many web sites include navigational maps to help users during browsing. Therefore, in our opinion, the characterization of navigational maps is a key issue during development of Web applications. Using navigational maps, developers can obtain a global view of the whole application that can help them in the development process. In addition, the presence of navigational maps can help users of Web sites to find the desired information much quicker.

Taking into account the development process of Web applications and the nature of the type of software, requires one to distinguish between two distinct stages in their development [18]. One stage focuses on conceptualization and prototyping, while the other pays attention to design and development. At conceptualization stage, the application is represented by a set of abstract models that convey the main components of the envisioned solution. At the design stage, conceptual schemas are transformed into a lower-level representation, closer to the needs of implementation, but still independent of the actual content of the information base.

This paper presents the Pipe notation [3] and uses it to characterize the navigational map of the travel agency's case study at the conceptualization stage. The paper also shows the mapping of the travel agency navigational map in terms of Pipe notation to an UML-Conallen class diagram [10] at the design stage. Although Pipe was originally intended as a hypermedia notation [1], its use in Web engineering projects has demonstrated its applicability as a tool to characterize navigational maps. In addition, due to the abstraction level of Pipe notation, it's possible to map it to specific design notations.

The paper is organized as follows. Section 2 briefly describes Pipe notation. Section 3 provides the Pipe characterization of the travel agency's navigational map. Section 4 maps Pipe characterization to UML diagrams that use Conallen's extension. Section 5 compares Pipe with other approaches. Finally, in section 6 conclusions and future work plans are presented.

## 2. Pipe notation

Pipe is a formalized graphical notation for the characterization of hypermedia applications. A description of the notation can be found in [2] whereas the formalization can be found in [3]. Pipe is conceived to be used in domains not easily representable in terms of entities/classes and their relations. Consequently, Pipe is focused on characterizing: (i) the units of information as individuals and relations among them; (ii) the user interface of the application, and (iii) the relations between the layer of contents and the layer of user interface to provide navigational access to the elements of the application.

Pipe provides three types of diagrams to characterize the above mentioned information: (i) a *contents graph*, which characterizes the information elements of the application and their relations; (ii) a *navigational schema*, which characterizes the graphical user interface of the application; and (iii) a set of *canalization functions*, which relate the contents graph and the navigational schema. This section briefly describes these elements.

### 2.1 Contents graph

The Pipe contents graph models the units of information in the application and the way they are related. It only considers individual units of information, anchors, links and a relationship function that captures all the semantic relations between the units. The Pipe contents graph uses the following modeling elements:

- *Static contents*. Units of information that exist prior to any interaction by the user.
- *Static anchors*. Anchors whose destinations are static contents.
- *Static links*. Relations between static anchors.
- *Dynamic contents*. Units of information generated by user's interaction [21].
- *Dynamic anchors*. Anchors whose destinations are dynamic contents.
- *Dynamic links*. Relations between dynamic anchors. In Pipe, dynamic links are characterized using a *generation function g* [3] specific for each application, but defined considering a common signature. Therefore, *g* function acts like the implementation of object-oriented interfaces and is used in Pipe to model dynamic behavior in the hypermedia applications at a conceptual level.

Figure 1 depicts the visual elements of the contents graph.

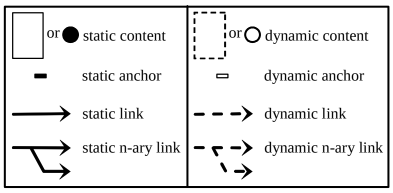


**Figure 1. Visual elements of the contents graph**

The Pipe graphical notation is fully formalized [3]. The formalization of the contents graph is made in terms of the *relationships function r*. This function has an extensional definition for static anchors, and an intensional definition, in terms of the generation function *g*, for dynamic anchors. In such a way, function *r* acts as a *black box* that "hides" the real nature of the links (static or dynamic).

### 2.2 Navigational schema

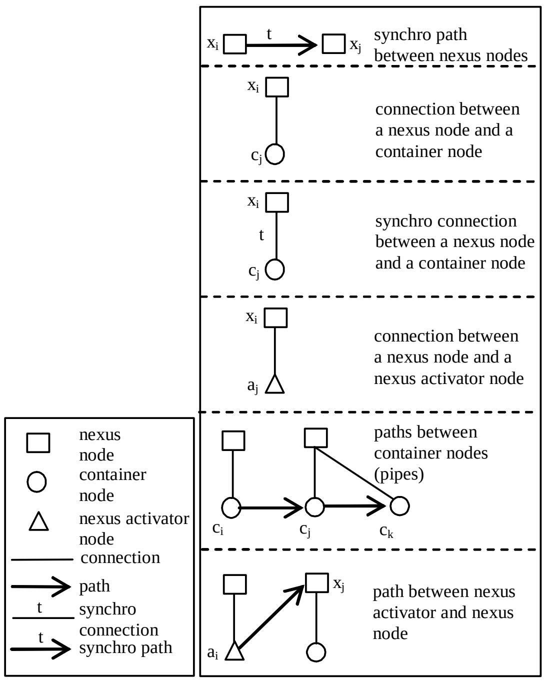
The Pipe navigational schema makes an important abstraction of the graphical elements that compose a graphical user interface (GUI) and the navigational relations that appear among the elements of the GUI in hypermedia applications. The Pipe navigational schema uses the following modeling elements:

- *Nexus nodes*, which represent the *windows* that contain the rest of elements of the GUI.
- *Container nodes*, which represent the *panes* that are inside the windows. They contain the contents (static or dynamic) of the contents graph.
- *Nexus activator nodes*, which represent the *buttons* that are inside the windows. The only event associated to this type of buttons is the activation of another window.
- *Connections*, which represent the way to relate a container node or a nexus activator node to its nexus node (i.e. they represent relations of aggregation between windows and their panes and buttons).
- *Paths*, which represent the navigational relations that exist between container nodes in the GUI. In this way if a path exists between a container node $c_1$ and a container node $c_2$, it is possible to display in $c_2$ the destination of a link with origin in a content that is being

displayed in $c_1$. Paths are also called *pipes* because they are responsible for *canalizing* (interpreting) the relations (links) established between the units of information (contents) in the navigational schema (GUI).

- *Synchro connections*. Which represent time-activated connections.
- *Synchro paths*. Which represent time-activated paths established between nexus nodes.

Figure 2 depicts the visual elements of the navigational schema and the allowed relationships between them.



$x_i$ t $x_j$ synchro path between nexus nodes
$x_i$ $c_j$ connection between a nexus node and a container node
$x_i$ t $c_j$ synchro connection between a nexus node and a container node
$x_i$ $a_j$ connection between a nexus node and a nexus activator node
$c_i$ $c_j$ $c_k$ paths between container nodes (pipes)
$a_i$ $x_j$ path between nexus activator and nexus node
nexus node
container node
nexus activator node
connection
path
t synchro connection
t synchro path

**Figure 2. Visual elements of the navigational schema and allowed relationships between them**

## 2.3 Canalization functions

The Pipe canalization functions relate the elements of the contents graph to the elements of the navigational schema. In this way several user interfaces can be used with the same units of information and semantic relations. Likewise, the same user interface can be used with different units of information and/or semantic relations. There are two canalization functions in Pipe:

- *Content assignation function*. This function assigns units of information (contents) to container nodes (panes).
- *Canalization function*. This function assigns relations between units of information (links) to paths/pipes among container nodes (panes).

Colors and visual patterns are used in Pipe to characterize the canalization functions.

## 3. Conceptualization of the travel agency

The description provided for the travel agency in the workshop's specification is focused on the functional behavior of the use cases in the system. Therefore, using a UML-based approach [7][16], use cases, activity, classes, interaction, component and deployment diagrams can be provided.

Although this specification is not focused on the description of the navigational map of the application, its navigational map can be easily characterized. Next sections describe this navigational map using the Pipe notation.

### 3.1 Contents graph

Pipe notation was intended to characterize hypermedia applications with domains not easily characterizable in terms of class or entities and their relationships [3]. Consequently, Pipe has focused on the characterization of individual units of information and their semantics relations [13]. As a result, the modeling primitives of Pipe are appropriate to characterize the heterogeneous relations established among individual *pages* of Web applications [10] (e.g. HTML [27] or JSP [9] pages).

Note that in the case study of the travel agency an important number of homogeneous elements and relations can be characterized using classes and their relations. Due to the origins of our approach, this information about classes and relationships cannot be represented in Pipe. Instead, UML class diagrams can be used to complement Pipe notation at the conceptualization stage. The integration of UML class diagrams and Pipe contents graph is made using the generation function *g*. The description of this integration process is out of the scope of this paper. In this section we provide the Pipe characterization of the contents accessed by the user during the browsing of the application, omitting the UML characterization. This Pipe characterization serves as the navigational map for the application. Section 4 provides the mapping of this Pipe characterization to UML diagrams at the design stage. In these UML diagrams the structure of classes and their relations could also be provided. Figure 3 shows the contents graph of the application that is indeed the navigational map of the travel agency.

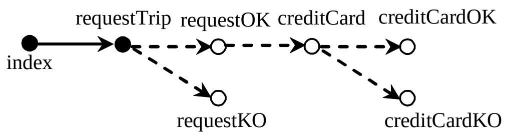


**Figure 3. Contents graph/navigational map of the travel agency**

The navigational map of Figure 3 describes a static index (`index`) which is linked to a static form that picks up data related to the requested trip (`requestTrip`). This form dynamically links [21] with the page that shows the result of the search for the trips in the travel agency. If the search fails (`requestKO`), the user must use the `index` to continue browsing the application. If the search obtains positive results (`requestOK`), the user can navigate to a form where his/her credit card data can be introduced (`creditCard`). Then these data are validated and independently of the result of the validation (`creditCardOK` or `creditCardKO`) the user must use the `index` to continue browsing the application. In this figure, the information about anchors [3] is omitted.

Note that in this contents graph, HTML pages are the characterized *contents*. Although *flights*, *hotels*, *clients*, etc. are the contents of the application from a data-modeling point of view, the HTML pages browsed by the user are the real contents of the application from Pipe's point of view. This is a *perverted* interpretation of Pipe, because in Pipe, the contents of the hypermedia applications are those heterogeneous units of information and their relations that are not easily representable in terms of classes/entities and their relations (e.g. a text that links with a sentence, and a sentence that links with the contextual meaning of their words [2]). In other words, because in Pipe there are no classes, individual elements are the contents of the application. The use of Pipe to characterize navigational maps of Web applications forces the interpretation of HTML pages as Pipe contents, being the real contents of the Web application represented in terms of UML class diagrams or entity-relationships [20] diagrams. These diagrams can be provided at conceptualization stage in order to complement Pipe diagrams and in order to fully characterize the contents of the travel agency.

In any case, Pipe contents are nearer to elementary items extracted from a data source and composed in different manners within pages than to pages displayed to users. Pipe notation is intended to be used at conceptualization stage. Therefore, in Pipe the content `requestOK` characterizes to the trips extracted from the database of trips according to customer's constraints. If the content `requestOK` has to include headers and footers with common information, a *frame-based approach* (see discussion in section 3.2 and 4.2) can be the best choice in Pipe. If the content `requestOK` has to include information dynamically extracted from different databases (possibly in different computers) the function *g* is the responsible of characterizing this process. For the sake of conciseness, the characterization of the function *g* is outside of the scope of this paper. In any case, such a characterization can be found in [3].

## 3.2 Navigational schema

Pipe navigational schema, characterizes the user interface of the application. Although the specification of the travel agency does not include the characterization of any user interface we are going to suppose a simple one. The application has a screen with two vertical regions. The left region is reserved to show the index of the application, while the right region is reserved to show the rest of contents of the application.

The terms *screen* and *region* are used instead of terms such as *frameset* and *frame* because, in our opinion, at conceptualization stage these details can be omitted. Indeed, during design this simple and abstract conception of the user interface could be refined, providing that the basic interaction behavior is preserved. For example, some designs can decide to use frameset and frames to represent screens and regions. Other designs can decide to aggregate the index to every content of the application (except the index to itself, of course), omitting the use of frameset/frames. Figure 4 depicts the Pipe characterization of this simple user interface.

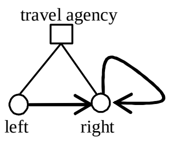


**Figure 4. Navigational schema/user interface of the travel agency**

According to this characterization, there is a window with two *panes* (or a screen with two regions). The arrow (*pipe*) between panes `left` and `right` means that the "destination contents" of the links with the origin in "displayed contents" of the `left` pane are displayed in the `right` pane. The arrow (pipe) between `right` pane and itself means that the "destination contents" of the links with the origin in "displayed contents" in `right` pane are displayed in the same pane.

We are aware that windows, panes and buttons are the basic elements of a graphical user interface. Regarding specific widgets (e.g. group or check boxes), Pipe does not provide explicit elements to characterize them. If wished, some visual content elements (e.g. group or check boxes elements) could be defined in order to be included in the container nodes of the user interface.

### 3.3 Canalization functions

Canalization functions represent the navigational interpretation of contents and relations in terms of the user interface of the application. Figure 5 depicts this information.

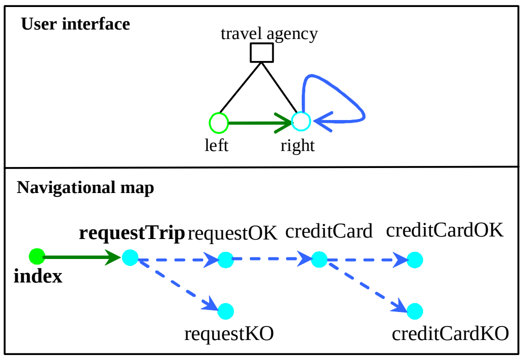


**Figure 5. Relations between user interface and navigational map**

Note that according to the colors of Figure 5, the content `index` is assigned to the `left` pane, while the rest of the contents are assigned to the `right` pane. In addition, the link between `index` and `requestTrip` is *canalized* by the pipe between `left` and `right`. The rest of the links are canalized by the pipe between the `right` and itself. Therefore, when the user selects the anchor in the content `index` (that is displayed in the `left` pane) that gives access to the content `requestTrip`, the content `requestTrip` appears in the `right` pane. In the same manner, if the user selects the anchor (i.e. *submit* button) in the content `requestTrip`, the content `requestOK` (or `requestKO`) appears in the `right` pane.

### 3.4 Browsing semantics

Browsing semantics describes the manner in which the information is to be visualized and presented [19]. Pipe includes a formalized browsing semantics [3] which describes applications' state, while the user is browsing.

Figure 6 describes travel agency's state, while the user is browsing it. Figure 6 (a) depicts the initial state where the context `index` is displayed in the `left` pane, and the content `requestTrip` is displayed in the `right` pane (the boldface in Figure 5 indicates that these contents are the default contents of these panes). Figure 6 (b) depicts the state of the application after the user requests a trip, and the application returns some options. Figure 6 (c) depicts the state of the application after the user decides to fill-in the form with the data of the credit card. Finally, Figure 6 (d) depicts the state of the application after the system validates the data of the credit card.

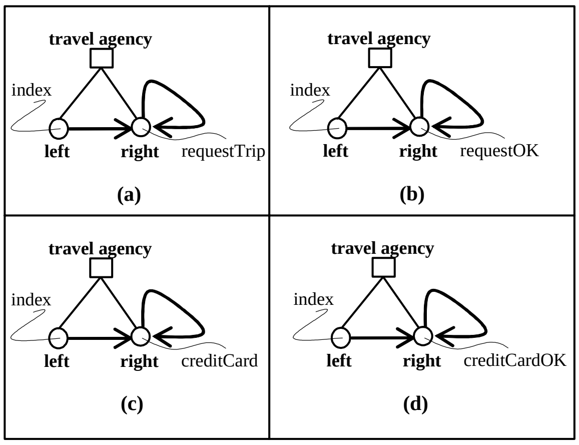


**Figure 6. State of the application while being browsed**

Note that the information provided in Figure 6, besides the information provided in Figure 5, can be used to indicate to the users the state of the application in terms of the navigational map and the user interface of the application.

## 4. Design of the travel agency

The design of the travel agency is represented in terms of the Conallen´s extension to the UML for Web applications [10]. The key point of Conallen´s extension is the principle of *separation of concerns*. According to this principle, there are in a Web application *client pages* (e.g. HTML pages) and *server pages* (e.g. JSP pages) that are represented in the UML by stereotyped classes and are related by navigated associations [10]. In addition, Conallen also provides a UML characterization of the structures of frames and framesets.

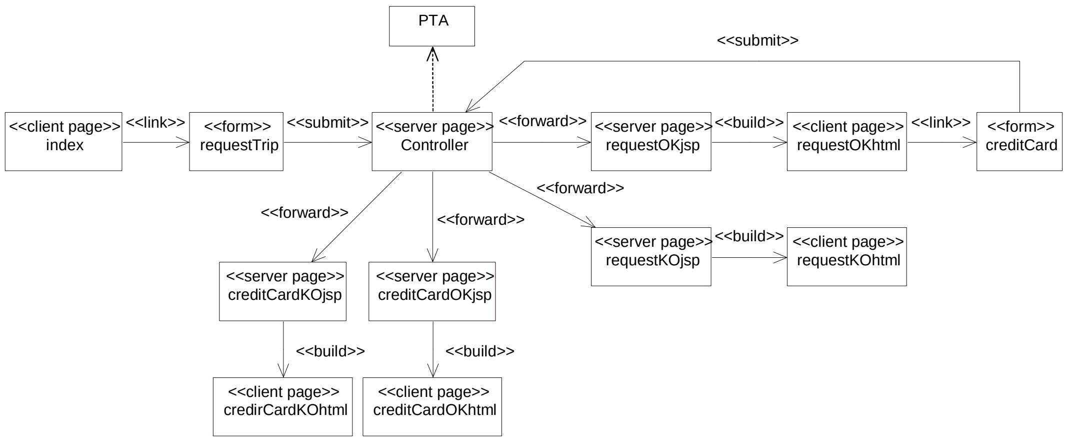

PTA
<<submit>>
<<client page>> index
<<link>>
<<form>> requestTrip
<<submit>>
<<server page>> Controller
<<forward>>
<<server page>> requestOKjsp
<<build>>
<<client page>> requestOKhtml
<<link>>
<<form>> creditCard
<<forward>>
<<forward>>
<<forward>>
<<server page>> requestKOjsp
<<build>>
<<client page>> requestKOhtml
<<server page>> creditCardKOjsp
<<server page>> creditCardOKjsp
<<build>>
<<build>>
<<client page>> credirCardKOhtml
<<client page>> creditCardOKhtml

**Figure 7. UML-Conallen navigational map of the travel agency**

This section provides the UML representation of the Pipe diagrams, taking into account the Conallen's extension.

## 4.1 Navigational map

Pipe content's graphs representing navigational maps can be smoothly mapped onto UML Conallen's extension during the design stage. Figure 7 shows the translation of the contents graph (Figure 1) for the present travel agency case study. Note that a *Model 2* (or *Model View Controller*) architecture [23] is chosen.

In this figure all the information, included in the specification about the computational behavior of the travel agency, is hidden within the class `PTA` (Personal Travel Assistant). In other words, the class `PTA` is a facade [5] of the application. For example, when the class `Controller` receives the information of the form `requestTrip`, the `Controller` delegates in the class `PTA` and, taking into account its response, the `Controller` forwards the control to the JSP pages `requestOKjsp` or `requestKOjsp` to generate the output for the user in terms of HTML pages (`requestOKhtml` or `requestKOhtml`). Note that the *Struts* API [26] could be used to implement this design. In addition, an *Enterprise Java Bean*-like [8] design could be used to implement the `PTA` facade and the rest of classes. Also, it should be noticed that the `PTA` facade gives the opportunity to model the actual data elements of the application and to incorporate the business logic associated with these elements. Because the paper is focused on navigational maps, the classes behind the facade are not provided. Although these classes constitute a substantial part of the application, their concrete structure and behavior is not relevant for characterizing the navigational part of this application.

## 4.2 User interface and final application

As previously mentioned, a frame-based or a non-frame based implementation of the regions could be provided at the design stage. If a frame-based implementation is chosen, Figure 8 describes its UML characterization in terms of Conallen's extension.

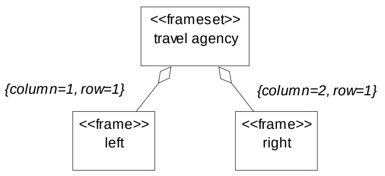


**Figure 8. UML-Conallen user interface of the travel agency**

In this figure, and in terms of Conallen's extension [10], the `left` frame could be omitted, since it is not a target of any link, but is included to provide a better characterization of the user interface. If a non-framed implementation is chosen, then all the classes (except the class `index`) must include an aggregation relation with the class `index`.

Finally, Figure 9 depicts the relations among the contents and the user interface, or in other words, the final application in terms of the notation defined in [10]. In this figure the Pipe canalization functions are used to assign contents to frames (e.g. content `index` is assigned to the frame `left`) and to interpret content links into the frame structure level (e.g. the link between `index` and `requestTrip` is targeted by the frame `right`).

## 5. Related work

At present, there exists an array of related work in the domains of hypermedia and Web engineering.

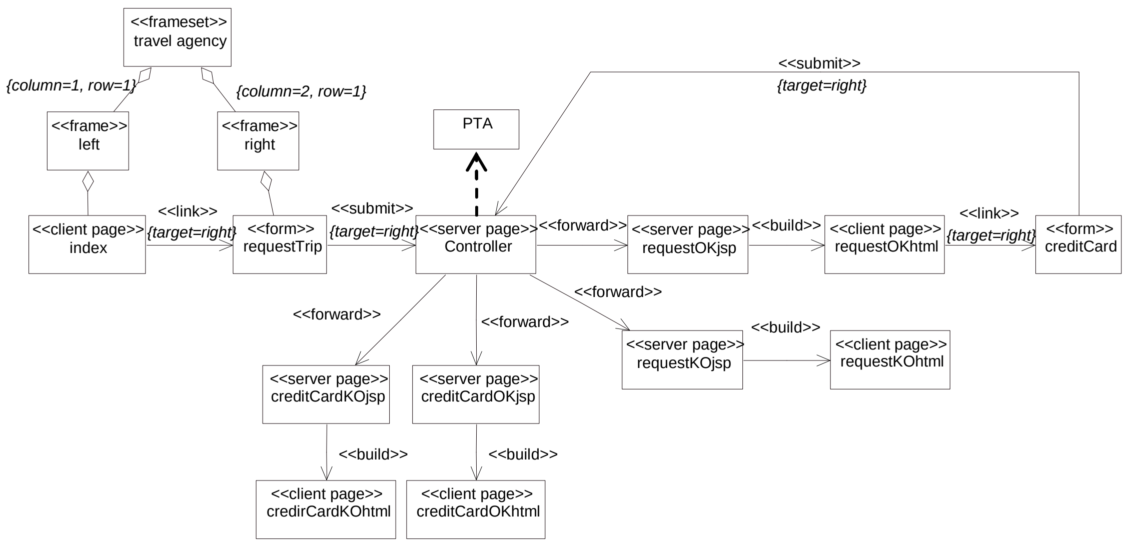


**Figure 9. UML-Conallen design of the travel agency**

*Hypertext Design Model* (HDM) [6], *Object-oriented Hypertext Design Model* (OOHDM) [4] and *Relationship Management Model* (RMM) [25], are some of the most relevant design notations in the hypermedia domain. All of them provide modeling primitives to characterize the classes or entities of the applications and their semantic relations. Because HDM and RMM characterize the applications from a data-model point of view none of them provides an explicit characterization of the navigational map of the application. OOHDM provides several modeling diagrams to characterize the navigational aspects of the application in terms of its classes and their relationships (i.e. navigation classes and navigational schema). In our opinion, the characterization of navigational maps in terms of classes and their relationships makes it harder for the users to visualize the navigational structure of the application since it requires the knowledge of object-oriented notations.

*Object-oriented hypermedia* (OO-H) [11], *UML-based Web Engineering* (UWE) [16], *Web Modeling Language* (WebML) [24], and *Web Site Design Method* (WSDM) [17] are some of the most relevant design notations in the Web engineering domain today. All these notations use some or other sort of diagram to characterize the navigational interpretation of the structural diagrams: OO-H uses *navigation access diagrams*, UWE uses *navigation diagrams*, WebML uses *navigation specifications* and WSDM uses *navigation tracks*. Because the modeling philosophy is similar in these approaches and in OOHDM (assimilating classes and relations), the conclusions derived for OOHDM are also applicable to these approaches.

Regarding Conallen's extension to UML, the use of stereotyped classes permits the presence of individual elements (e.g. the client page `requestTrip` of Figure 8) that can be used to characterize navigational maps for Web applications. In our opinion, Conallen's extension is well – suited for design, because it permits definition of the object-oriented behavior of dynamic applications, but is not especially suited for the conceptualization stage. Indeed, it is more complicated to use it for conceptualization, since it ties-in more this stage with the design stage than the Pipe does. In addition, in our opinion, the Pipe characterization of Figure 3 of the travel agency's navigational map is clearer than the navigational map that could be derived from the UML characterization of Figure 7 because: (i) it is not necessary the knowledge of object-oriented notations to understand Figure 3; (ii) Figure 3 hides architectural details included in Figure 7 (e.g. the presence of a Model 2 architecture); and (iii) Figure 3 hides technical details regarding the presence of client and server pages. Therefore, and as our experience has demonstrated, Pipe notation is simpler to understand by customers at the conceptualization stage [1][2][3].

Regarding Pipe notation scalability, in our opinion, it is similar to the scalability of visual notations (e.g. Conallen's extension). If a great amount of contents appear in the diagram, the separation in several subdiagrams can be the best choice.

*Dialog Flow Notation* (DFN) [12] and *State WebCharts* (SWC) [14] are focused on the characterization of the navigation in Web applications. DFN represents the dialog flow within an application as a directed graph of states connected by transitions, and SWC uses statecharts to describe the navigation between documents. To some extent, both approaches characterize Conallen’s separation of concerns. DFN uses *masks* and *actions* [12] while SWC uses *static*, *transients* and *dynamic* states [14]. In Pipe the separation among client and server pages is not considered because the dynamic processing is hidden behind the generation function *g*. In addition DFN and SWC use the same diagram to characterize the structural and navigational models, while in Pipe these models are represented using contents graph and navigational schema diagrams.

## 6. Conclusions and future work

The characterization of navigational maps is a key issue during development of Web applications. Using navigational maps, developers can obtain a global view of the whole application that can help them during the development process. In addition, the presence of navigational maps can help users of Web sites to find the desired information quicker.

This paper presents a use of the Pipe notation to characterize navigational maps for Web applications (hypermedia or non-hypermedia). Pipe contents graph can be used to characterize the navigational map for Web applications at conceptualization stage, thus omitting computational design details included in Conallen’s extension. In addition, Pipe contents graph permits the explicit definition of Web application’s navigational map instead of deriving it from modeling primitives defined in terms of classes/entities and their relations.

Using Pipe navigational schema and canalization functions it is possible to provide a presentational characterization of the navigational map in terms of screens and regions. This characterization, besides the Pipe browsing semantics, could be used to characterize the application’s state during its browsing.

Moreover, as demonstrated in this paper, the generation of UML-Conallen class diagrams using Pipe contents graph is a straightforward task. Although the translation from Pipe to UML diagrams presented in this paper does not include any complex detail, we are aware that a systematic and well-defined conversion procedure between Pipe and UML (in both directions) has to be defined. Indeed, at present, our main research effort is made in this direction.

Being focused on the representation of heterogeneous units of information and their heterogeneous relations, the Pipe notation lacks the modeling primitives for characterizing domains that can be easily represented in terms of classes/entities and their relations. For this purpose, UML class diagrams or entity-class diagrams can be used in combination with Pipe diagrams. In this paper we have omitted such a characterization because we have focused on the provision of a navigational map for the travel agency instead of characterizing its data and their relations.

Currently, we are developing a CASE tool to support the Pipe notation. Future work includes the automatic translation from Pipe notation to Conallen's extension (in both directions) to alleviate the transition from conceptualization to the design stage. In addition, we are working on the automatic generation of HTML pages and object-oriented codes from Pipe diagrams. UML-Conallen diagrams derived from Pipe diagrams could be used to generate such pages and codes.

## 7. Acknowledgements

The Spanish Committee of Science and Technology (TIC2001-1462, TIC2002-04067-C03-02 and TIN2004-08367-C02-02) has supported this work.

## 8. References

[1] A. Navarro, B. Fernández-Manjón, A. Fernández-Valmayor, and J.L Sierra, J.L. “Formal-Driven Conceptualization and Prototyping of Hypermedia Applications”. In proceedings of *FASE 2002-ETAPS 2002* Grenoble, April, 2002, pp. 308-322.

[2] A. Navarro, B. Fernandez-Manjón, A. Fernández-Valmayor, and J.L. Sierra, “The PlumbingXJ Approach for Fast Prototyping of Web Applications”. *Journal of Digital Information 5, 2, special issue on Information Design Models and Processes*, 2004.

[3] A. Navarro, A. Fernández-Valmayor, B. Fernandez, and J.L. Sierra, “Conceptualization, Prototyping and Process of Hypermedia Applications”. *International Journal of Software Engineering and Knowledge Engineering 14, 6, special issue on Modeling and Development of Multimedia Systems*, 2004, pp. 565-602.

[4] D. Schwabe, G. Rossi, and S.D.J. Barbosa, “Systematic Hypermedia Application Design with OOHDM”. In *Proceedings of the Hypertext 96,* Washington DC, USA, March, 1996, pp. 116-128.

[5] E. Gamma, R. Helm, R. Johnson and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[6] F. Garzotto, P. Paolini, D. and Schwabe, HDM. A Model-Based Approach to Hypertext Application Design. *ACM Transactions on Information Systems, 11, 1,* 1993, pp. 1-26.

[7] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*, Addison-Wesley, 1999.

[8] Java Technology. Enterprise Java Beans Technology http://java.sun.com/products/ejb/index.jsp

[9] Java Technology. Java Server Pages Technology. http://java.sun.com/products/jsp/

[10] J. Conallen, "Modeling Web Application Architectures with UML". *Communications of the ACM 42, 10*, 1999, pp. 63-70.

[11] J. Gómez, C. Cachero, and O. Pastor, "Conceptual Modeling of Device-Independent Web Applications". *IEEE MultiMedia 8*, 2, 26-39.

[12] M. Book, and V. Gruhn. "Modeling Web-Based Dialog Flows for Automatic Dialog Control". In *Proceedings of the. 19th IEEE International Conference on Automated Software Engineering (ASE 2004)*, Linz, Austria, 2004, pp. 100-109.

[13] M. Thüring, J.M. Haake, and J. Hannemann, "What's Eliza Doing in the Chinese Room? Incoherent Hyperdocuments - and How to Avoid Them". In *Proceedings of the Hypertext 91*, Texas, 1991, pp. 161-177.

[14] M. Winckler, M., and P. Palanque. "StateWebCharts: a Formal Description Technique Dedicated to Navigation Modelling of Web Applications". In *Proceedings of the International Workshop on Design, Specification and Verification of Interactive Systems (DSVIS'2003),* Funchal, 2003.

[15] N. Koch, and A. Kraus, "Towards a Common Metamodel for the Development of Web Applications". *In proceedings of International Conference on Web Engineering 2003, ICWE 2003*, 2003, pp. 497-506.

[16] N. Koch, H. Baumeister, R. Hennicker, and L. Mandel, "Extending UML to Model Navigation and Presentation in Web Applications". *In Modeling Web Applications in the UML Workshop, UML2000*, York, England, October 2000.

[17] O.M.F De Troyer, and C.J. Leune, "WSDM: A User Centered Design Method for Web Sites". *Computer Networks* 30, 1-7, pp. 85-94.

[18] P. Fraternali, "Tools and Approaches for Developing Data-Intensive Web Applications: A Survey". *ACM Computing Surveys 31, 3,* 1999, pp. 227-263.

[19] P.D. Stotts and R. Furuta, "Petri-Net-Based Hypertext: Document Structure with Browsing Semantics". *ACM Transactions on Office Information Systems, 7, 1, 1989,* pp. 3-29.

[20] P.P. Chen, "The Entity-Relationship Model - Toward a Unified View of Data". *ACM Transactions on Database Systems 1, 1,* 1976, pp. 9-36.

[21] R.C. Bodner, M.H. and Chignell, "Dynamic hypertext: querying and linking". *ACM Computing Surveys 31, 4es,* 1999.

[22] S. Abrahão and O. Pastor. "Measuring the Functional Size of Web applications" *International Journal of Web Engineering and Technology 1, 1, 2003*, pp. 5-16.

[23] S. Brown et al. *Professional JSP. 2nd Edition*. Wrox Press, 2001.

[24] S. Ceri, P. Fraternali, and A. Bongio, "Web Modeling Language (WebML): a modeling language for designing Web sites". *Computer Networks 33, 1-6*, 2000, pp. 137-157.

[25] T. Isakowitz, E.A. Stohr, and P. Balasubramanian, "RMM: A Methodology for Structured Hypermedia Design". *Communications of the ACM 38, 8,* 1994, pp. 34-44.

[26] The Apache Software Foundation. Struts. http://struts.apache.org/

[27] W3C. HTML 4.01 Specification, 1999. http://www.w3.org/TR/html4/