Improving the Web Application Design Process with UIDs

PATRÍCIA VILAIN
Depto. de Informática e Estatística, Universidade Federal de Santa Catarina, Brazil
vilain@inf.ufsc.br
and

DANIEL SCHWABE[?]
Depto. de Informática, PUC-Rio, Brazil
schwabe@inf.puc-rio.br

In this paper, we present a diagrammatic technique to represent the information exchange during the interaction between the user and the application called User Interaction Diagram (UID). UIDs have proven to be a valuable technique to gather requirements since they describe the exchange of information in a high level of abstraction, without considering specific user interface aspects and design details. UIDs can be incorporated into the web application design process. During the requirements gathering, they can be used to represent the requirements; during the conceptual design, they serve as a basis for the synthesis of a preliminary class diagram using heuristic guidelines. Moreover, during the navigation design, additional heuristic guidelines can be used to specify partial context schemas from UIDs.

Categories and Subject Descriptors: D.2.2 [Design Tools and Techniques] *Modules and interfaces*, D.2.1 [**Requirements/Specifications**] *Languages, Methodologies*H5.2 [**Information Interfaces and Presentation**]: User Interfaces - *User-centered design*; *Interaction styles*; *Theory and methods*; H.5.4 [**Hypertext/Hypermedia**] *Navigation, Theory, User Issues*
General Terms: Design, Experimentation, Human Factors
Additional Key Words and Phrases: Human-computer interaction, Hypertext, Design

1. INTRODUCTION

A major distinguishing feature of Web applications from traditional information-system type of applications is navigation. Whereas both organize information into meaningful items, Web applications allow the user to navigate through these items using its navigational structure.

Traditional software development methods reflect this difference – they do not usually address navigation as an abstraction on its own, which should be separated from interface operations. This need for separation has been recognized since the early 90's, first with HDM (Hypermedia Design Model) [GARZOTTO et al. 1991, GARZOTTO et al. 1993], followed by several others such as RMM, OOHDM, WebML, etc… [SCHWABE AND ROSSI 1998, LANGE 1994, ISAKOWITZ et al. 1995, CERI et al. 2000, HENNICKER AND KOCH 2000]. All these methods include specific navigation models [ROSSI et al. 1999].

The focus of this work is on modeling the interaction – i.e., information exchange - between the application and the user, differently than most existing methods mentioned above.

[?] Daniel Schwabe was partially supported by a grant from Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPQ – Brazil.

The Unified Modeling Language (UML) [BOOCH et al. 1999] suggests using use cases to model the interaction with the user during requirement elicitation[1]. Text-based techniques such as scenarios and use cases, can produce complete and detailed specifications, and are typically easier to understand by non-technical people. On the other hand, such specifications become large, ambiguous and overloaded in the case of larger applications, making them less understandable and more difficult to manage.

The existing diagrammatic techniques, such as sequence diagrams, collaboration diagrams, statechart diagrams, activity diagrams, data-flow diagrams and even essential use cases are not able to focus on the interaction and at the same time satisfy all of these requirements. The UML specifies the realization of use cases using sequence, collaboration and statechart diagrams [BOOCH et al. 1999]. Such diagrams, however, are better suited for system design (and implementation) than for requirements gathering, since they unnecessarily force early design decisions, such as object identification.

To fill this gap, we present a diagrammatic modeling technique called User Interaction Diagram (UID), focusing exclusively on the information exchange between the application and the user [VILAIN et al. 2000]. The ultimate goal is to support the design process of Web applications in incremental levels of detail, starting with the gathering of requirements. At this initial stage, the technique should present the interaction information in a structured, detailed and differentiated way (between user and application); it should allow easy communication between the designer and its client and between the designer and implementers; and it should support traceability from requirements to posterior design steps, all the way to implementation. Figure 1 shows the use of UIDs throughout the web application process.

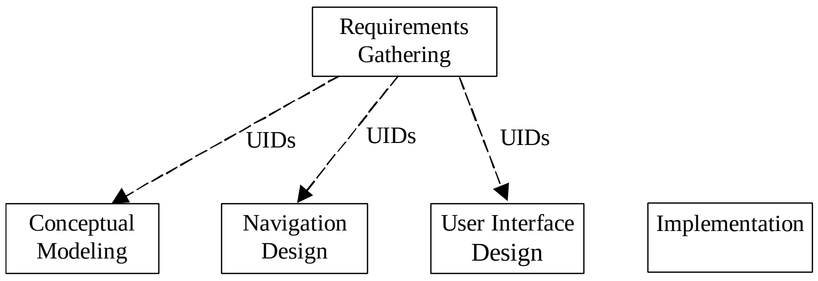


Figure 1. UIDs in the Application Process.

This work is organized as follows. Section 2 presents the diagrammatic technique used to represent the interaction between the user and the web application: UID. Section 3 shows how to use UIDs in the requirements gathering. Section 4 shows how to obtain a preliminary conceptual model from UIDs. Section 5 presents the incorporation of UIDs in the navigational modeling phase of Object Oriented Hypermedia Design Method OOHDM [GÜELL et al. 2000]. Finally, sections 6 and 7 present the future works and conclusions.

This work presents UIDs taken from a case study of an online CD shop, which involved 6 participant users and resulted in 14 use cases and 14 UIDs.

[1] It is important to distinguish between the concepts of interaction used in this work and the one used in the object-oriented models, such sequence and collaboration diagrams used in UML [BOOCH et al. 1999]. In UML, interaction is seen as the exchange of messages among objects defined in the application implementation.

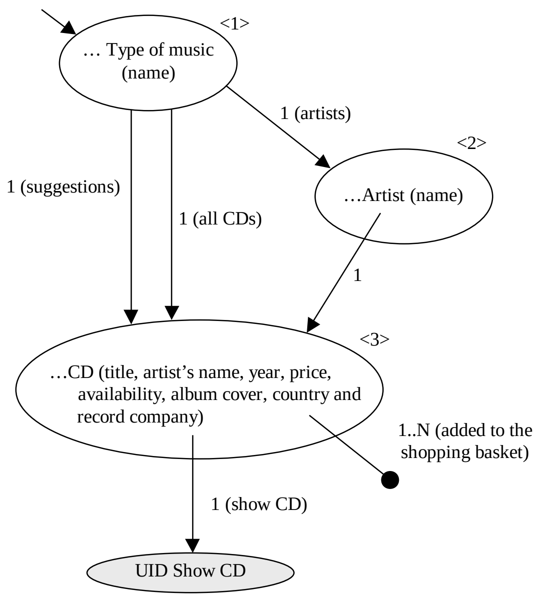
2. UIDs

This section presents a new diagrammatic notation to represent the exchange of information between the user and an application called User Interaction Diagram (UID). In fact, UIDs can be seen as a diagrammatic description of the information exchanged that is textually described in a use case.

UIDs consider neither user interface aspects nor navigation aspects. Moreover, we try to define a UID notation as simple as possible, so the user can understand it easily. This notation has been derived over several versions, incorporating feedback received from the meetings held with the case studies users and with some graphics and interface designers.

UIDs are composed by a set of interaction states connected by transitions. Interaction states represent the information exchanged between the user and the application, while transitions represent the change of the interaction focus from one interaction state to another. We can say that an interaction state is the interaction focus when the information inside it represents the information that is being exchanged between the user and the application. Transitions are triggered, in general, by information entered or selected by the user. Next, we present one example of UID; the appendix shows the complete UID notation.

Figure 2 shows the UID defined to represent the interactions in the task *Selection of a CD based on the type of music*. In this task, the user selects a type (genre) of music and the system returns a set of CDs that matches this type. From this set, the user adds to the shopping basket the CD s/he wants to buy later.



<1>
… Type of music (name)
1 (artists)
<2>
…Artist (name)
1 (suggestions)
1 (all CDs)
1
<3>
…CD (title, artist's name, year, price, availability, album cover, country and record company)
1..N (added to the shopping basket)
1 (show CD)
UID Show CD

Figure 2. Selection of a CD based on the type of music: UID.

In the initial interaction state, the system presents a set of types of music. During this interaction state, the user must select one type of music followed by the choice of one of the following options: *artists*, *all CDs*, *suggestions*. If the user selects the option *artists*, then the interaction state that shows the set of artists (for the chosen type of music) becomes the focus of interaction. Following this interaction state, the user must select one artist and the system shows the set of CDs related to this artist. Once again, the set of

CDs related to the chosen artist is presented in a separate interaction state since it depends on the artist selected.

If the user selects the option *suggestions*, then the system shows the set of CDs that are recommended for the chosen type of music. If the user selects the option *all CDs*, then the system shows all CDs under the chosen type of music.

Supposing that a set of CDs has at least one element, it can be represented by ellipsis in front of the set name (*…CD* in the example) or explicitly by its cardinality (*1..N CD*). As a CD is represented by a set of related information items, it is represented by a structure. For each CD shown by the system, the following data items are given: *title, artist's name, year, price, availability, album cover, country, and record company*.

From the interaction state <2>, it is possible to select 1 to N CDs, and the option of adding them to the shopping basket (*1..N (add to the shopping basket)*) is also available. It is also possible, from this interaction state, to select one CD with the option *show CD*, which transfers the focus of the interaction from the current UID to the UID *Show CD*.

3. REQUIREMENTS GATHERING

In this section, we describe how we can use UIDs during the requirements gathering. UIDs are defined from scenarios or use cases. The definition of scenarios and use cases is outside the scope of this work; any method of specification can be used in this phase. For example, we can use the method Scenario Construction Process [LEITE et al. 2000] or Inquiry Cycle Model [POTTS et al. 1994].

We specify a UID for each scenario or use case defined. In order to obtain a UID from a given scenario or use case, the sequence of information exchanged between the user and the system must be identified and organized into interactions. Identifying this information exchange is crucial since it is the basis for the definition of the UID.

In the following, we present some guidelines to map a scenario or use case to a UID.

Step 1. We start by analyzing the scenario or use case to identify what data items are exchanged between the user and the system, which are generally represented by nouns. It is also important to identify which data items are given by the user and which are returned by the system.

Figure 3 shows the use case *Selecting a CD based on a given artist's name* with the information given by the user underlined and the information given by the system in italic.

> The user enters the **name of the artist**. S/he can inform the **year of the CD** being sought if s/he wants to. The system returns ***a set with the names of the artists*** matching the value entered; if only one name of artist matches, the ***set of CDs by this artist*** is shown. The user selects **the artist of interest**. The system returns ***a set with the CDs by this artist***. For every CD found, ***the title, name of the artist, year, price, availability, and an image of the album cover*** are shown. The user can access more information about a CD if s/he wants (use case Show CD). If the user wishes to buy more than one CD, s/he selects the CD(s) and adds it (them) to the shopping basket to perform the purchase later (use case Purchase).

Figure 3. Information Exchanged in a Scenario.

Step 2. After identifying the information exchanged, we have to split the data items into interaction states. Data items are placed in the same interaction state, unless they depend on the previous data items or an option must be selected first; in such cases, they are placed in another interaction state. However, information given by the system that is followed by information given by the user can both be placed in the same interaction state. Next, we try to place the remaining data items in this interaction state, and so on. The scenarios and use cases called from the scenario or use case being analyzed are placed in interaction states representing the calling of other UIDs.

In the example, we begin with the *name of the artist* data item, which is placed in the interaction state <1> of the UID. The next data item exchanged, the *year of the CD*, given by the user, does not depend on the previous data item (*name of the artist*), so it can be placed in the same interaction state. Since the next data item exchanged, *set with the names of the artists*, depends on *the name of the artist* and *the year*, it is placed in an interaction state <2>, and so on.

Step 3. The data items given by the user and returned by the system must be distinguished. Mandatory data items given by the user are placed into rectangles with a continuous border and the optional data items are placed into rectangles with a dashed border. The data items returned by the system are placed directly in the interaction, without rectangles around them. Data items that are associated, defining a structure, are placed inside parenthesis after the name of the structure. Sets are represented by their cardinality before the name of the data item or structure. Sets with cardinality 1..N may be represented by ellipsis before its name.

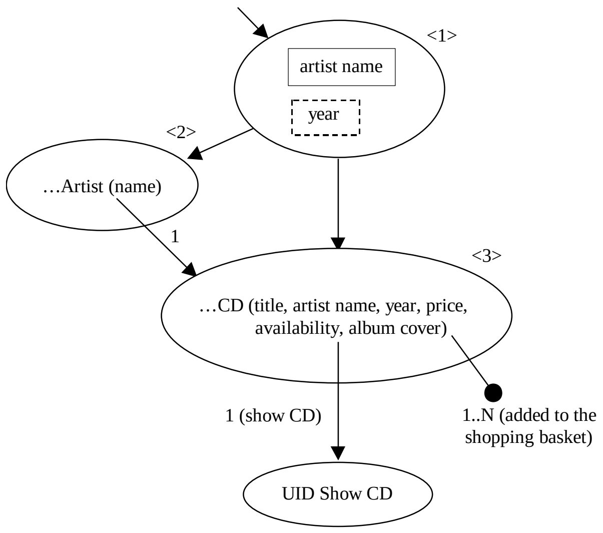
Step 4. The interaction states are connected by transitions. It is possible to connect one interaction state to two or more interaction states, thus representing several alternative succeeding interaction states. In this case, the information entered by the user determines which of the alternative interaction states will be the next focus of the interaction. If the change of focus of the interaction is the result of an element selection, the number of selected elements is attached to the arrow, and the source of this arrow is the set from which the selected elements are taken. The initial interaction state always has a transition without source.

Step 5. The operations executed by the users are identified and represented by options. They are generally represented in the scenarios or use cases by verbs. Options are represented in the diagram as labels attached to transitions. If after the selection of the option, the focus of the interaction does not change, then the option can be represented attached to a line with a bullet at the end.

Step 6. Non-functional requirements are not graphically represented in UIDs. Although use cases do not specify these requirements, sometimes a non-functional requirement can appear. In this case, it should appear attached to the UID as textual note.

Figure 4 represents the complete UID.

After defining the UIDs, the designer interacts with each user to validate the use cases and their corresponding UIDs. In order to do this, the designer shows the use cases and their respective UIDs to verify if the users agree with them; an agreement process is also established to deal with conflicting user requirements. It is important to explain the notation of the UIDs to the users; in our experience, users learn it without difficulty and are able to interact with the designer to express their understanding of the task being modeled with the UID.



<1>
artist name
year
<2>
…Artist (name)
1
<3>
…CD (title, artist name, year, price, availability, album cover)
1 (show CD)
1..N (added to the shopping basket)
UID Show CD

Figure 4. Complete UID for the Scenario in Figure 3.[2]

4. CONCEPTUAL MODELING

In this section, we present some guidelines to determine a preliminary class diagram based on a UID, aimed at identifying aspects related to the data requirements. The main guidelines to obtain a class diagram from UIDs are the following:

Guideline 1: For each UID, define a class for each structure.

Before applying the second guideline, it is necessary to analyze all UIDs. The classes defined from each UID are added to the class diagram.

Guideline 2: For each data item A, returned by the application or given by the user, appearing in each UID, define an attribute according to the following:

For each class, verify that the attribute corresponding to the data item depends on that class. In this verification, we apply the question[3]:

> *Question 1: Given an instance of the class X, is it possible to obtain the value of the attribute A?*

If the answer is affirmative for only one class, then the data item becomes an attribute of that class.

If the answer is affirmative for two or more classes, then the following question is applied to the combination of these classes:

> *Question 2.a: Given an instance of the class X, is it possible to obtain the information (attribute values) of a single instance of the class Y?*
>
> *Question 2.b: Given an instance of the class Y, is it possible to obtain the information (attribute values) of a single instance of the class X?*

The data item becomes an attribute of the class from which it is not possible to obtain the information of an instance of the other class.

Consider the interaction state shown in figure 5. We have to find out what classes the attributes corresponding to the data items depend on. Suppose we have identified the classes *CD*, *Artist*, and *Type of Music* from the first guideline. We start the analysis with the data item *title*. First, we verify if the attribute *title* depends on the class CD (*CD*

[2] Numbers in brackets are not part of the notation.

[3] These questions actually establish the functional dependency between class instances and data items.

represents the structure that the data item *title* is related to) applying the question 1: "Given a CD, is it possible to obtain the CD title?" The answer to this question is affirmative. The same question is asked for classes *Artist* and *Type of Music* ("Given an artist, is it possible to obtain the CD title?" and "Given a type of music, is it possible to obtain the CD title?"). However, the answers to these questions are negative, since for each artist there are several CD titles, and for each type of music, there are numerous CD titles. Therefore, the data item *title* becomes an attribute of the class *CD*.

In the same way, we analyze each of the remaining data items related to the structure *CD*.

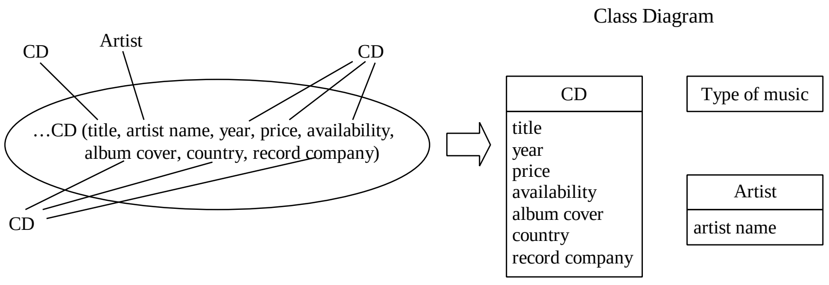


Figure 5. Identifying Attributes.

Adjustments of Class and Attribute Names

After identifying the classes and the attributes, sometimes it is necessary to solve some conflicts among class and attribute names before continuing the process.

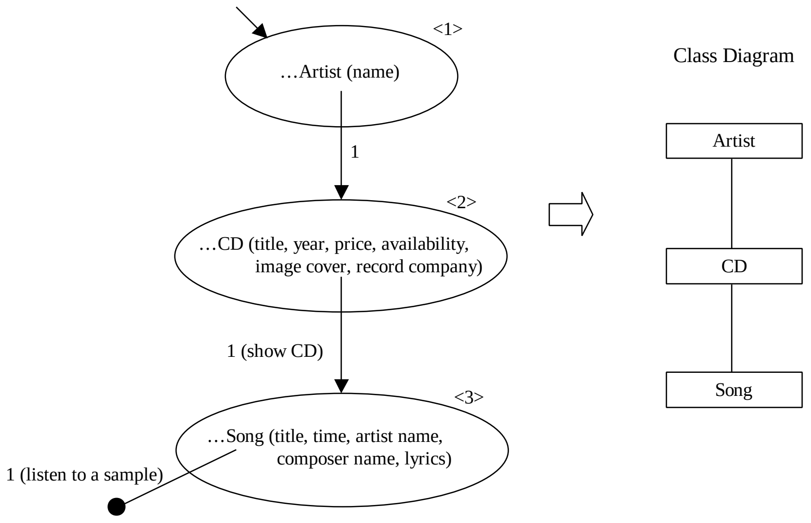
Guideline 3: For each attribute *a*, whose data item appears in a structure that does not correspond to the class of *a*, if there is another attribute *b*, whose data item appears in the same structure, but belongs to a different class from that of *a*, and it is possible to obtain the information of an instance of the class of *a* from an instance of the class of *b*, create an association between the class that contains *a* and the class that contains *b*. Verify if the resulting association is semantically correct (i.e., if it makes sense in the domain being modeled).

Guideline 4: For each UID, for each structure s_1 that contains another structure s_2, create an association between the classes that correspond to the structures s_1 and s_2. If the class was eliminated, use the class that has replaced it. Verify if the resulting association is semantically correct.

Guideline 5: For each interaction state transition (represented by an arrow) in each UID, if there are different classes representing the source interaction state and the target interaction state, define an association between the corresponding classes. A class represents an interaction state if its corresponding structure is the most meaningful information presented in the interaction state or if the data items corresponding to its attributes are the most meaningful information presented. It is also necessary to verify if this association is semantically correct.

Consider the UID presented in Figure 6. According to the guideline 1, the class *Artist* is defined from the structure *Artist* in the interaction state <1>, the class *CD* is defined from the structure *CD* in the interaction state <2>, and the class *Song* is defined from the structure *Song* in the interaction state <3>.

The first transition connects an interaction state related to the class *Artist* to an interaction state related to the class *CD*. Therefore, according to the guideline 5, we create an association between the classes *Artist* and *CD*. In the same way, applying the guideline 5 on the second transition, we create an association between the classes *CD* and *Song*.



<1>
…Artist (name)
1
<2>
…CD (title, year, price, availability, image cover, record company)
1 (show CD)
<3>
…Song (title, time, artist name, composer name, lyrics)
1 (listen to a sample)
Class Diagram
Artist
CD
Song

Figure 6 - Identifying Associations

Guideline 6: For each selection of an option that has two or more different sources, define an association between the classes representing them. The sources of the selection may be a structure that is represented in the association by its class or a data item, which is represented in the association by the class that has the corresponding attribute. It is also necessary to verify if this association is semantically correct.

Guideline 7: For each UID, the options that appear in a UID are user interface operations. During the definition of the preliminary class diagram, they can be defined or not as operations of the classes. Despite the fact that they are defined as operations in the class diagram, these operations can be modified during the design phase.

Final Adjustments

At the end of the process, do the necessary adjustments in the resulting class diagram. The most important are:

1. Identify the generalizations. We do not have any guidelines to define generalizations from UID. Therefore, it is necessary to analyze all attributes and associations among the classes to identify the generalizations.
2. Define the missing association cardinalities. Even when using the guidelines, by the end of the process we are not likely to find out the cardinalities of all associations.
3. Verify that there is a class corresponding to an association between two classes. If there is, the class representing the association must be eliminated and replaced by an association.

5. NAVIGATION DESIGN

The UIDs defined during the requirements gathering are used, in conjunction with scenarios, use cases and the conceptual model, as the basis to the navigation design. The main task is to identify the meaningful sets of navigation items that make up the navigation space, and their corresponding access structures. In the following, we present some guidelines to map UIDs to the navigation model of OOHDM. Their purpose is to identify aspects related to the navigation requirements. These guidelines are applied after defining the navigation classes of OOHDM.

Although formulated for OOHDM, these guidelines may also be used for methods that identify navigation during conceptual modeling, in which case they should be applied after the class diagram has been synthesized. If navigation is represented in conjunction with the class diagram, these guidelines could be adapted to map UIDs to directed associations, instead of to context schemas.

5.1 Mapping UIDs to Context Schemas

In [GÜELL et al. 2000], it is suggested that information of the sets must be analyzed according its importance to the task, which is gleaned from scenarios, use cases, users, and clients. In addition to this information, design patterns can also be used.

5.1.1 Mapping Sets of Structures

Sets of structures returned by the application, presented in UIDs, can be accessed by users through indexes, contexts or lists. A list means that all instances of the class corresponding to the structure are shown together, without navigation among them.

Mapping sets of structures into Indexes

A set of structures is mapped into an index of the context schema if the task requires that the elements of the set be compared among themselves, allowing the user to select the desired element. In general, a set mapped into an index presents the following characteristics:

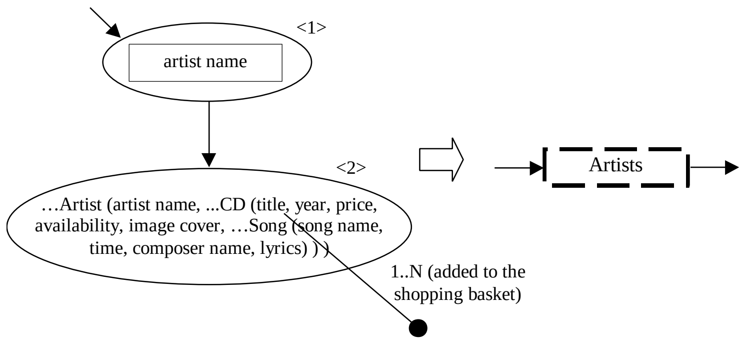
1. The structure of the set does not present all data items (attributes) of the class representing it;
2. The set is the source of the outgoing transition from the interaction state, and the target is another interaction state;
3. The structures presented in the interaction state of the set and the interaction state connected to it by the transition are different. If they present the same structure, the source structure presents all attributes and the target structure does not present all attributes.

Figure 7 presents the mapping for the set of structures in the UID *Selection of CDs and Songs based on artist's name* into indexes in the context schema. As several sets of structures are nested and belong to the same interaction state, i.e., interaction state <2>, not all common characteristics used to map a set of structures into an index can be applied. In this case, it is important to analyze the task corresponding to the UID. In the example, since the set of artists, presented in interaction state <2>, is used only to obtain a set of CDs, it is mapped to an index (*Index Artists*). Similarly, since the set of CDs and the set of songs will not have anchors to other elements, they are not mapped, in this step, into indexes.

Mapping Sets of Structures into Contexts

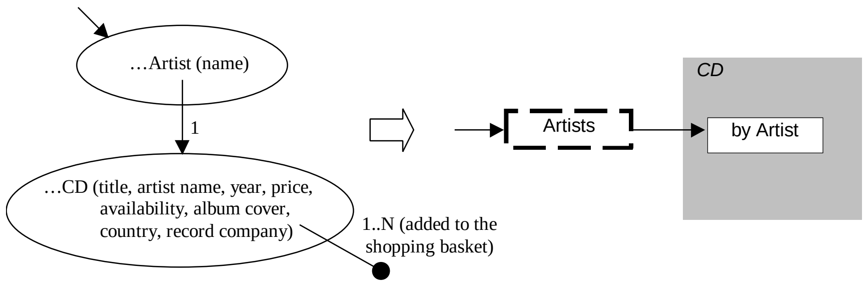
A set of structures is mapped into a context of the partial context schema if the task requires that the information of a single element be accessed by the user. In general, the set mapped to a context has the characteristic that the structure of the set presents all data items corresponding to the attributes of the class representing the structure.

Figure 8 presents a simple UID, *Selection of a CD based on artist's name*, and the mapping of its sets of structures into contexts. Consider that the set of structures *Artist* was mapped into the *Index Artist.* Supposing that the task corresponding to the UID requires that the user accesses all information about each CD, and that there is no navigation from this information, then the set of CD is mapped into a context. This context is called *CD by Artist* because it is accessed from the *Index Artist*.



<1>
artist name
<2>
…Artist (artist name, ...CD (title, year, price, availability, image cover, …Song (song name, time, composer name, lyrics)))
1..N (added to the shopping basket)
Artists

Figure 7. Mapping sets of structures into Indexes. The *Artists* set in state <2> is mapped onto the *Artists* index.



…Artist (name)
1
…CD (title, artist name, year, price, availability, album cover, country, record company)
1..N (added to the shopping basket)
Artists
CD
by Artist

Figure 8. Mapping sets of structures to Contexts. The CD set of structures is mapped onto the *CD* by artist context.

A context may or may not have an index associated to it, depending on the task corresponding to the UID. If during *Mapping Sets of Structures into Indexes*, an index to the context is defined, then it is not necessary to define a new index. Otherwise, the designer has to identify the most significant data items that will make up the index items.

Suppose that the task corresponding to the UID presented in the figure 8 requires that after selecting an artist, an index showing all his/her CDs be shown, instead of presenting his/her first CD. In this situation, an index to access the context *CD by Artist* is defined, and thus all CDs are accessed through this index.

Mapping Sets of Structures into Lists

A set of structures is mapped into a list if the task requires that the set elements to be compared among themselves, but be accessed simultaneously.

5.1.2 Mapping Single Structures

A single structure (a structure that appears alone, i.e., a structure that does not belong to a set) is mapped to a context of the partial context schema if the task requires that the information of an element be accessed by the user. This element must be part of a set of elements that have a common characteristic, making up a context.

If the interaction state, where the single structure appears, presents an incoming transition from an interaction state that also presents information about the same element, then the single structure is not mapped to a context.

If the interaction state, where the single structure appears, presents more than one incoming transition with different options, then a context for each transition is defined.

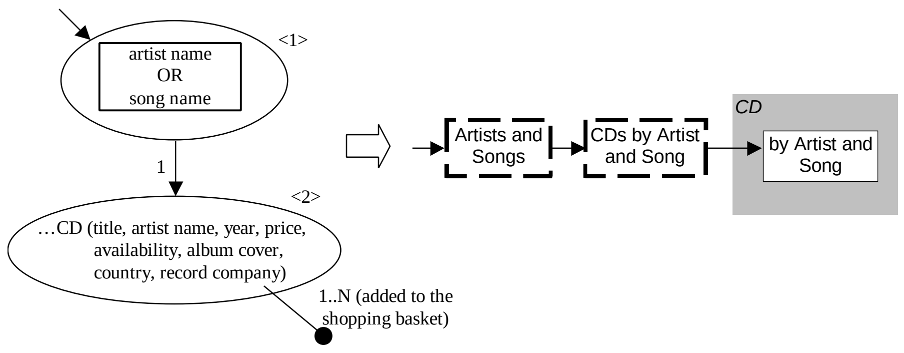
5.1.3 Mapping Data Entries

A data entry is mapped to an index of the partial context schema if the task requires a query that returns a specific element of the context. In general, data entries mapped onto an index present the following characteristics:

1. they appear in the first interaction states;
2. in the next interaction states, the system returns a single structure, several data items, or a set of structures that is mapped to an index.

If the interaction state of a data entry is followed by an interaction state of an index related to the same information, then the data entry and the index are represented by a unique index.

Figure 9 presents the mapping of the data entries in the UID *Selection of a CD based on artist's name and song's name* to indexes of the context schema. Consider that the artist name and the song name are used, in the task corresponding to this UID, to search and select a set of CDs. Thus, the data entries are mapped to an index (*Index of Artists and Songs*). This mapping considers that the set of CDs, presented in the interaction state <2>, was mapped, previously, to a context (*CDs by Artist and Song*) associated to an index (*Index of CDs by Artist and Song*).



<1>
artist name OR song name
1
<2>
…CD (title, artist name, year, price, availability, album cover, country, record company)
1..N (added to the shopping basket)
Artists and Songs
CDs by Artist and Song
CD
by Artist and Song

Figure 9. Mapping Data Entries to Indexes

5.2 Mapping the UID Options

If an option that appears in the UID is associated to an operation, then this operation can be specified by a sequence diagram [BOOCH et al. 1999].

The UID *Selection of a CD based on the Type of Music*, shown in the figure 2, presents 5 options associated to its transitions: *artists*, *all CDs*, *suggestions*, *show CD*, and *add to the shopping basket.* The first four options do not have any operation associated to them. The fifth option, *add to the shopping basket*, has an operation associated to it. The specification of this operation can be given by a sequence diagram, as shown in figure 10.

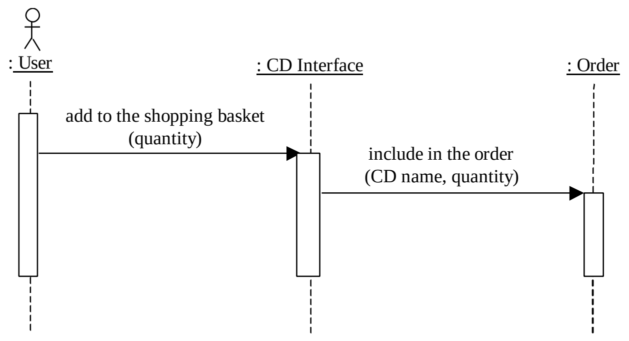

: User
: CD Interface
: Order
add to the shopping basket (quantity)
include in the order (CD name, quantity)

Figure 10. Specifying an Operation (add to the shopping basket).

5.3 Mapping Interactions with No Navigation

If we must specify the internal functionality of an application, we can also specify interaction states that do not present navigation among them as process using sequence diagrams [BOOCH et al. 1999]. If we do not need to specify the internal functionality, then UIDs are enough for modeling these processes and the application functionality associated to them.

Figure 11 shows the UID *Checkout* and figure 12 shows a sequence diagram that specifies the message exchange among internal objects of the application during the corresponding task.

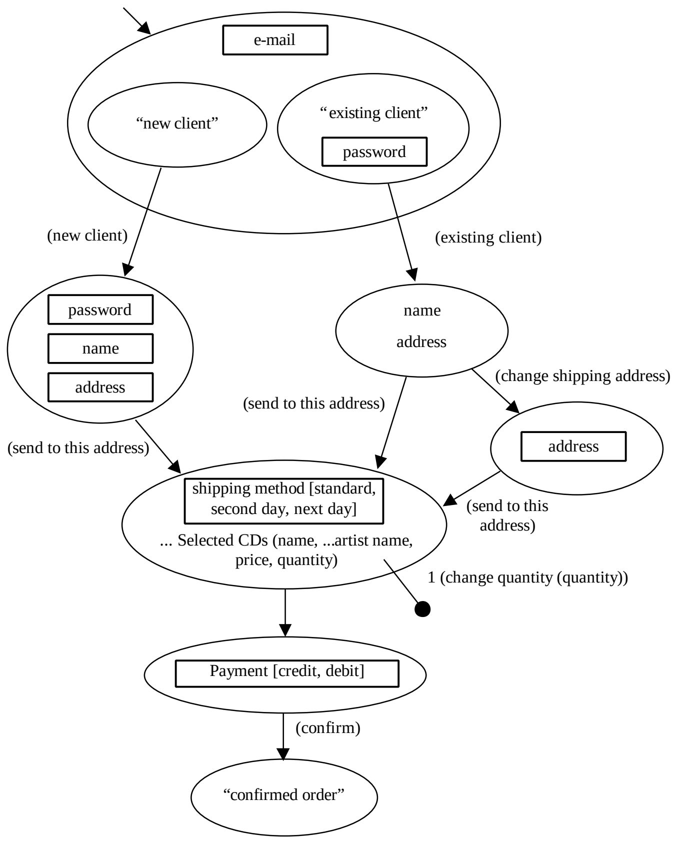
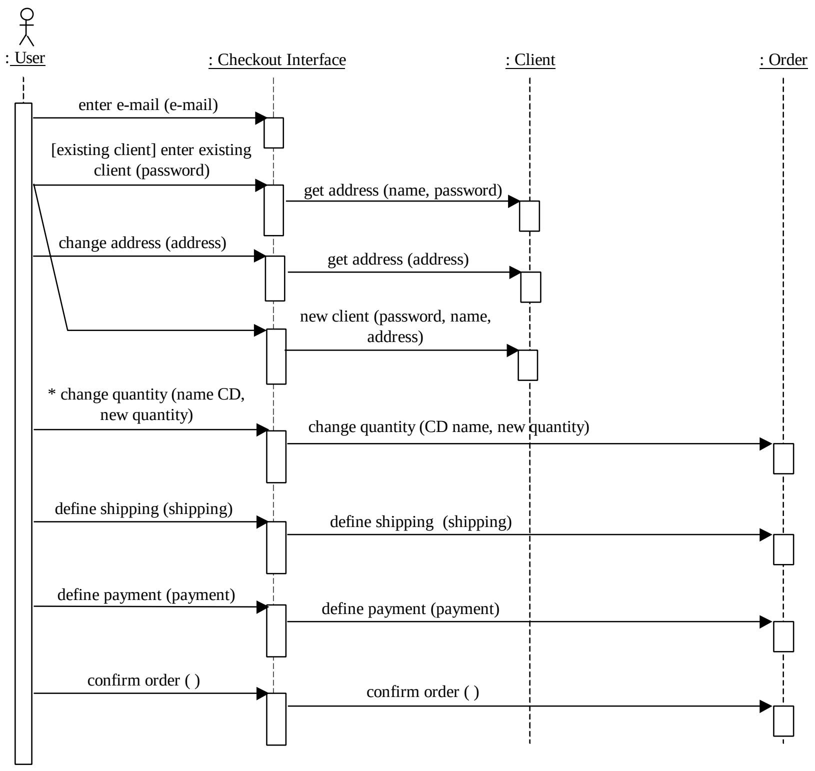


Figure 11. Checkout: UID



: User
: Checkout Interface
: Client
: Order
enter e-mail (e-mail)
[existing client] enter existing client (password)
get address (name, password)
change address (address)
get address (address)
new client (password, name, address)
* change quantity (name CD, new quantity)
change quantity (CD name, new quantity)
define shipping (shipping)
define shipping (shipping)
define payment (payment)
define payment (payment)
confirm order ( )
confirm order ( )

Figure 12. Specifying a Process

5.4 Synthesis of the Final Navigation Context Schema

After all partial navigation schemas have been defined, the designer starts an incremental process of unification among these partial schemas. This unification tries to identify identical contexts, or contexts that can be "unified" into a more general context, such that the unified ones can be "special cases". In addition, navigation among contexts belonging to different partial schemas must also be analyzed, and those deemed useful by the users are added to the final schema. Details of this process may be found in [GÜELL et al. 2000].

6. CONCLUSION

We have presented a diagrammatic technique that represents the interaction between the user and the application called User Interaction Diagram (UID). In addition, to represent the information exchange between the user and the application, UIDs are the basis for the conceptual modeling and the navigation design. In order to accomplish that, we have provided heuristic guidelines allowing the specification of the conceptual model and the navigation model of OOHDM, based on the UIDs.

UIDs can be used without concern over user interface and navigation aspects. Differently from the diagrams usually used to represent use cases (sequence diagrams, collaboration diagrams, state transition diagrams, activity diagrams, statecharts), UIDs specify all information that participates in the interaction between the user and the application, without consideration for design aspects.

It is important to point out that we observed, during the case studies, that users prefer to discuss the application requirements using UIDs instead of using textual descriptions. UIDs also facilitate the identification of partial context schemas, during the navigation design, because it is easier to find indexes and contexts from the interaction states of UIDs than from a textual description.

UIDs are also being used for the OOHDM [GÜELL et al. 2000]. In that method, a new phase was included: requirements gathering. This phase defines and uses UIDs. Moreover, the existing conceptual design and navigation design phases were modified to incorporate the guidelines used to define a preliminary class diagram and partial navigation contexts.

In spite of UIDs having been used only with OOHDM, they can also be used with other methods for the web application development. Some examples of them are the process of developing web applications [CONALLEN 2000] and the UML based methodology for hypermedia design [HENNICKER AND KOCH 2000]. Since UIDs are used in the first phases of the development cycle, its adjustment to other methods is easier. If a method does not present the requirements gathering phase, e.g. WebML (Web Modeling Language) [CERI et al. 2000], it can use any requirements gathering incorporating UIDs.

We are currently pursuing several enhancements and applications of UIDs:

? Development of heuristic guidelines for mapping UIDs to interface specifications;

? Introduction of personalization requirements;

? Utilization of UIDs as a first step in the elaboration of Web Application Design Frameworks;

? Utilization of UIDs as a basis for cost estimation for web application development projects.

REFERENCES

BOOCH, G., AND RUMBAUGH, J., AND JACOBSON, I. 1999. *The Unified Modeling Language User Guide*. Addison-Wesley, Reading, Massachusetts.

CERI, S., AND FRATERNALI, P., AND BONGIO, A. 2000. Web Modeling Language (WebML): a modeling language for designing Web sites. In *Proceedings of the WWW9 Conference*, Amsterdam, May 2000.

CONALLEN, J. 2000. *Building Web Applications with UML*. Addison-Wesley.

GARZOTTO, F., AND PAOLINI, P., AND SCHWABE, D. 1991. HDM - A Model for the Design of Hypertext Applications. In *Proceedings of the Hypertext'91*, San Antonio, 1991, 313-328.

GARZOTTO, F., AND PAOLINI, P., AND SCHWABE, D. 1993. HDM - A Model-Based Approach to Hypertext Application Design. *ACM Transactions on Information Systems*, 11, 1, 1-26.

GÜELL, N., AND SCHWABE, D., AND VILAIN, P. 2000. Modeling Interactions and Navigation in Web Applications. In *Proceedings of the Second Workshop on Conceptual Modeling and the WWW - ER'2000*, 2000, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 115-127.

HENNICKER, R., AND KOCH, N. 2000. A UML-based Methodology for Hypermedia Design. In *Proceedings of the UML2000 Conference*, York, England, October 2000, Lecture Notes in Computer Science, Springer, Berlin, 410-424.

ISAKOWITZ, T., AND STOHR, E., AND BALASUBRAMANIAN, P. 1995. RMM: A methodology for structuring hypermedia design. *Communications of the ACM*, 38, 8, 34-44.

LANGE, D. 1994. An Object-Oriented Design Method for Hypermedia Information Systems. In *Proceedings of International Conference on System Science*, Hawaii, 1994, 366-375.

LEITE, J.C.S.P., AND HADAD, G.D.S., AND DOORN, J.H., AND KAPLAN, G.N. 2000. A Scenario Construction Process. *Requirements Engineering*, 5, 38-61.

POTTS, C., AND TAKAHASHI, K., AND ANTÓN, A.I. 1994. Inquiry-Based Requirements Analysis. *IEEE Software*, 21-32.

ROSSI, G., AND SCHWABE, D., AND LYARDET, F. 1999. Web Application Models Are More than Conceptual Models. In *Proceedings of the First Workshop on Conceptual Modeling and the WWW - ER'99*, 1999, Springer, Paris, 239-252.

SCHWABE, D., AND ROSSI, G. 1998. An object-oriented approach to Web-based application design. *Theory and Practice of Object Systems (TAPOS)*, 207-225.

VILAIN, P., AND SCHWABE, D., AND SOUZA, C.S. de. 2000. A Diagrammatic Tool for Representing User Interaction in UML. In *Proceedings of the UML2000 Conference*, York, England, October 2000, Lecture Notes in Computer Science, Springer, Berlin, 133-147.

APPENDIX: USER INTERACTION DIAGRAM NOTATION

In the following, we present the notation used by UIDs to represent the interaction between a user and an application.

Data Item: represents single information that appears during the interaction. A data item can be followed by colon and the domain name. The domain is specified by the designer. If the domain is not specified, domain *Text* is assumed.

<data item> : <domain>

<data item>

Structure: a collection of information (data items, structures, data items set, or structures set) that is somehow related. Information set of a structure is specified between parentheses after the structure name. The information can be suppressed in the early versions of the UID.

<Structure> (<data item$_1$>, <data item $_2$>.. <data item $_{n>}$)

<Structure> ( )

Set: represents a set of data items or structures. The multiplicity of a set is represented by min..max in front of the data item or structure. The default multiplicity is 1..N and is represented just by a ellipsis (…).

…<data item>

… <Structure> (<data item $_1$>, <data item$_2$> .. <data item$_n$>)

1..5 <data item>

Optional Data: represents an optional data item, structure or text. An optional information is represented by the symbol "?", but it could also be represented by a set with multiplicity 0..1. In the case of an user entry, an optional data can be represented by a dashed rectangle.

<data item>?

<Structure> (<data item $_1$>, <data item$_2$> .. <data item $_n$>)?

<data item>

User Input: represents a data item or structure entered by the user. All information entered by the user is placed inside a rectangle.

<data item>

Enumerated User Input: represents a data item entered by the user from a list of values given by the system. The values given by the system appear between brackets, separated by commas. If more than one value is selected, the quantity is indicated in front of the data item.

<data item> [<option$_1$>, <option $_2$> .. <option $_N$>]

System Output: represents a data item or structure returned by the system. All information returned by the system is placed directly in the interaction state.

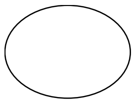
<data item>

<Structure> (<data item $_1$>, <data item $_2$> .. <data item $_n$>)

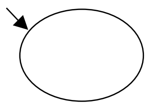
Text: represents a descriptive text, shown by the system that appears in the interaction.

“ <text> “

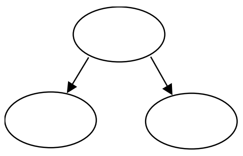
Interaction State: represents an interaction state of the interaction between the user and the system. The information given by the user and returned by the system is usually shown inside the ellipse. An interaction state can never be empty.



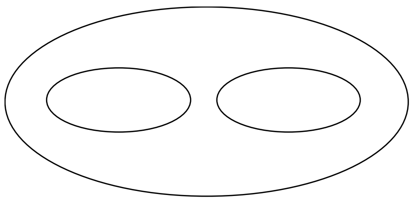
Initial Interaction State: represents the initial interaction state of the interaction between the user and the system.



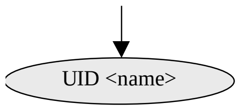
Interaction State Alternatives: this representation is used when there are two or more alternative outputs from an interaction state. The subsequent interaction state that will become the focus of the interaction depends on the information entered or option chosen by the user.



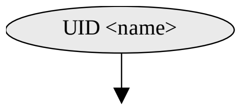
Interaction State Sub-States: this representation is used when two parts of an interaction state are exclusive. The exclusive parts are placed in different sub-states.



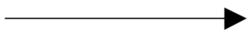
Calling of Other UID: represents that the interaction focus is transferred to other UID.



Calling from Other UID: represents that the interaction focus is transferred from other UID.



Interaction State Transition: represents that the target interaction state may become the new interaction focus after the system has returned the necessary information and the user has entered the required data in the source interaction state.



Transition with Selection of N Elements: represents that N elements must be selected before the target interaction state can become the interaction focus. The selected elements can be from different sets.

$$\xrightarrow{\text{<N>}}$$

Transition with Selection of the Option X: represents that the option X must be called before the target interaction state can become the interaction focus. The option name is represented between parentheses.

$$\xrightarrow{\text{(<X>)}}$$

When the selection of an option does not change the interaction focus, the transition can be represented by a line with a black bullet in one end.

$$\overset{\text{(<X>)}}{\relbar\joinrel\relbar\joinrel\relbar\joinrel\relbar\bullet}$$

Transition with Condition Y: represents that the target interaction state becomes the interaction focus if the condition Y is true. The condition is expressed using natural language.

$$\xrightarrow{\text{[<Y>]}}$$

Pre and Post Conditions: represents conditions that must be satisfied before and after occurring the interaction represented in the UID.

$$\boxed{\text{Pre-condition: <Y>}}$$

$$\boxed{\text{Post-condition: <Y>}}$$

Textual Notes: represents important information that cannot be graphically represented.