9

ViewNet - Conceptual Design and Modelling of Navigation

Jürgen E. Ziegler

Fraunhofer Institute IAO
Nobelstrasse 12, D-70569 Stuttgart
GERMANY
Juergen.Ziegler@iao.fhg.de

ABSTRACT This paper presents an approach to modelling navigation structures for complex, database-oriented applications. Navigation dialogues are first classified according to a number of underlying principles. The ViewNet technique is introduced for the conceptual design of the navigation structure of a system. ViewNets are diagrams composed of views and navigation links. Different view types are introduced on the basis of the mapping from conceptual objects to views and their role in the dialogue. These types are represented by specific graphical symbols which support the designer in getting a clear overview of the navigation structure and for optimizing dialogues according to the user's task.

KEYWORDS design methods, modelling techniques, navigation, dialogue structures, object-oriented user interfaces

1. INTRODUCTION

The main design issues in developing complex, database-oriented applications with graphical user interfaces are increasingly related to defining the dialogue structure of an application „in the large", i. e. to designing the overall structure of the user interface. Whereas design aids, e. g. in the form of styleguides are available for selecting and designing the standard interaction objects of a graphical user interface, the issue of designing suitable dialogue structures must usually be solved by the developer for each application specifically. In the best case, this process is currently supported by guidelines which are specific to an application or an organization. As yet, there is little methodological support for solving the complex design decisions involved. An appropriate conceptual design of the dialogue paths which make the functionality of a complex system accessible is , however, essential for the usability of the system, particularly as users are becoming more and more familiar with the basic handling of graphical user interfaces.

In order to better distinguish it from those parts of the dialogue which serve for manipulating the different interaction objects (like buttons, entry fields or lists), the term „navigation" shall be used here to describe the user's movement through the different views of a system. A „view" is defined here as a collection of elements which represent one or more underlying application objects or tasks in a coherent manner, e. g. in a window or screen form.

A number of techniques for modelling navigation sequences have been developed till now which basically aim at describing precisely the system's behaviour depending on the system's state and the user's input. Those techniques are typically based on state transition diagrams (see e. g. Denert 1977, Jacob 1986) or Petri nets (see e. g. Janssen 1993). Due to the detailed specification of trigger events, conditions and actions, however, they entail the danger of obstructing the view of the overall dialogue structure for the developer rather than making it transparent. Although some approaches have been proposed for designing the structure of the navigation e.g. in the field of hypermedia systems (see Nielsen 1990, Berk

Human–Computer Interaction: INTERACT'97 S. Howard, J. Hammond & G. Lindgaard (editors)
Published by Chapman & Hall ©IFIP 1997

& Devlin 1991, they are not well suited to database-oriented applications where the dialogue design has to take the cardinalities of object classes (many instances of the same type) into account.

On the other hand, the existing dialogue modelling techniques say little about the type and the properties of the views of which the navigation structure is composed (they generally only indicate the views'names). That is why they can be described as being rich in transition information but poor in state information. Information concerning the meaning and role of a view, however, is pertinent to the design of efficient and consistent dialogues, arguably more important during the early conceptual design stages than a detailed behavioural specification. Till now, a method has been lacking which takes both aspects into account and which particularly make the type and meaning of the views evident from which the navigation structure is built. This paper presents the ViewNet method which takes these requirements into consideration and supports the conceptual design of the navigation structures through a graphical modelling technique.

2. PRINCIPLES OF NAVIGATION DESIGN

Navigation dialogues can be structured according to a number of different principles. These principles relate to the respective predominant aspect of the user's task, such as the function to be performed, the object to be manipulated, or the selection of a task step from a complete business process to be accomplished. These different goal components translate into different requirements with respect to how the user will access the functionality needed for performing the task and the corresponding dialogue paths required. In general, navigation structures may be based either on a functional decomposition of the system, on the objects of an application and their relations or on arbitrary associations between different pieces of information. In the following, we will discuss some of these principles and their implications for the usability of the system.

Function-oriented navigation: The selection of a specific function or operation represents the starting point for the navigation (e. g. in a conventional hierarchical menu system). The actual data view becomes visible and can only be manipulated after one or several consecutive steps. This type of navigation is particularly suited to well-defined and repetitive tasks with little variability but has major drawbacks if the operation component of the user's task is not well defined at the outset or changes during the interaction.

Process-oriented navigation can be seen as an extension of the function-oriented principle and provides support for a complete set of tasks belonging to a specific work or business process. The system can control the status of the process and according to the situation, enable or disable the access to the objects and functions needed. Whereas early systems of this type used to force the user to perform a fixed sequence of steps, today's graphical interfaces allow a more flexible design . e. g. through visualizing the status of the process and currently available tasks in lists, task bars etc..

Object-oriented navigation represents the main paradigm in direct manipulation. It uses the objects of the application and of their semantic relations for the design of dialogue paths. Operations become only available when the object to be manipulated is selected and visible . Object-oriented navigation represents one of the basic principles of the graphical user interfaces and is particularly characterized by its high degree of consistency and flexibility.

Association-oriented navigation characterizes the typical navigation form in hypertext/hypermedia systems. The nodes of the navigation structure are represented by individual information units (in contrast to fixed object types with arbitrarily many instances in object-oriented navigation). The possible transitions are defined through the arbitrary associations between those information units.

These forms of navigation have different profiles concerning usability criteria such as efficiency, comprehensibility and flexibility. Combined forms are therefore frequently used in real applications.In the following section, we will focus on the different types of views which are involved in the composition of such navigation structures.

3. VIEW TYPES

In the ViewNet method, views are defined as logical collections of information elements, not as concrete visual representations. Views represent the underlying application objects, tasks or general information units wholly or in part and can be visualized in a coherent manner. Multiple views of an object are possible. Views can be composed of a hierarchy of subviews. In the following, we will introduce a classification of the different types of views relevant for user navigation and introduce a graphical notation for these types which forms the elements of a ViewNet representation.

Object views represent a single instance of a specific object class (e. g. 'customer') in different forms. Three

types of object views can be distinguished: the object reference view, which is often in the form of an icon (*icon view*), the *attribute view*, which represents all or some attributes of an object, and the *graphical view* which shows an arbitrary graphical representation of an object instance (e.g. as a map, a business graphic etc.).

In order to structure object-oriented dialogues, **collection views** are required which either show a partial or a complete collection of instances of a certain object class (e.g. as a list of instances) or comprise objects of different types as e.g. the objects on a desktop *(inhomogenous collection*). A special case of collection views are filtered collections through which the user has access to pre- or self-defined subsets of the data (e. g. all payable bills). In many cases, it will be advantageous if users can set up and manage such filter objects themselves at the user interface in a flexible way.

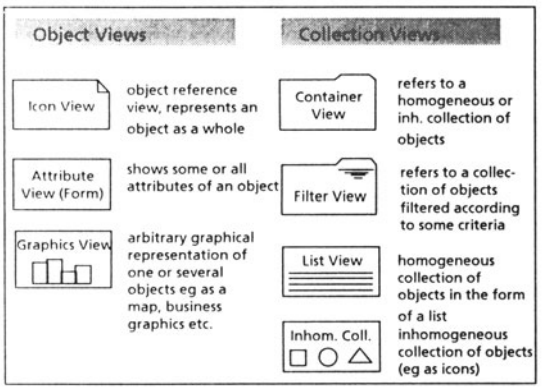

Figure 1: Object and collection views

Function views offer functionally oriented possibilities for navigation, as they are represented, for example, by menu screens or modal dialogue boxes (Figure 2). Object attributes may be shown in a function view but usually only as needed for the operation selected (e. g. search attributes in a search dialogue box). *Actor views* are a special form of a function view and represent the differente steps of a complex task or process the user needs to perform. Actor views help to guide the user through the steps of a complex process while maintaining application-specific dependencies and constraints. They can be realized in very different forms, e.g. as to-do lists, task lists or assistant windows which are becoming more and more popular in standard office products.

Information views, represented either as single nodes or clusters, can be used in association-oriented navigations. Examples for this can be found in hypertext-based help systems or generally in hypermedia systems. In contrast to the links between of object and collection views which represent 1:n or n:m relations in an underlying object model (or entity-relationship model), the relations involved in the navigation between information views are usually of a one-to-one type.

Figure 3 shows **composite views**. Typical aggregations which are often used in graphical interfaces like *master-detail views* or *notebooks*, are depicted with their own graphical symbol in order to get a comprehensive and intuitive overview of the dialogue structure. *Generic building blocks* allow to collapse the parts of a complete navigation substructure into a single element which can be parameterized with the object class accessed in this navigation. This way, for example, a search dialogue for instances of a class which consists of several views can be parameterized with the name of the class. Similar search dialogues for different classes (e. g. orders, customers, products) can then be represented by a single symbol with an additional indication of the respective class. This mechanims reduces the size of the diagram, improves clarity and supports a consistent development of the navigation structure.

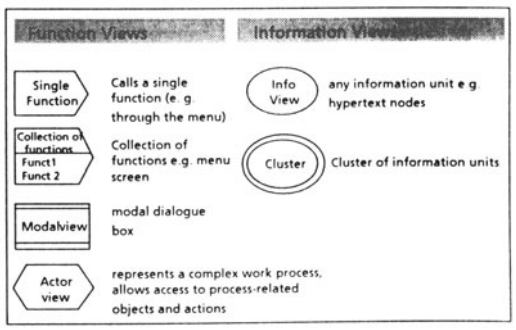

Figure 2: Dialogue and information views

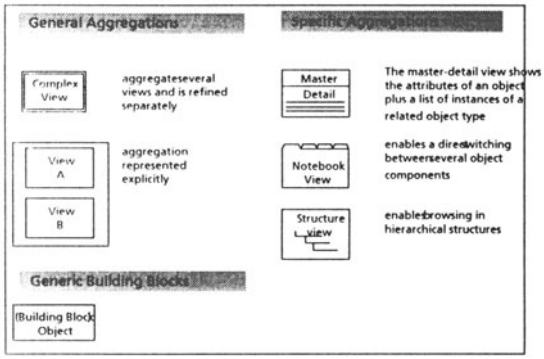

Figure 3: Composite views

4. MODELLING NAVIGATION WITH VIEWNETS

By using the view types defined and their graphical symbols, navigational structures can be conveniently represented in a diagrammatic form. In order to achieve a clear overview of the navigation, especially during the initial design steps, ViewNets in their simplest form model only whether a user can reach other views from a given node. Trigger events, conditions and actions are not represented at this stage. For a more detailed specification of the dynamic behaviour of the dialogue, however, the model can be extended in the form of dialogue nets, a specific Petri net representation for dialogues (Janssen, Weisbecker & Ziegler, 1993). For the purpose of designing the navigation at a conceptual level, it will in most cases be more appropriate to abstract from the details of the interaction.

Figure 4 shows a typical object-oriented navigation structure which might be used, for example, in the design of an order management system. The topmost level of the navigation (e.g. a graphical desktop) corresponds here to the drawing surface in order to simplify the model. Starting from an icon view of object collections (customers) on the desktop, the user can reach the attribute view of a particular object instance by opening a list of customers and selecting a specific instance or by a suitable search mechanism. Operations are made available locally in the attribute view e.g. as buttons or menu entries. The attribute view ‚Customer´ is represented here as a composite notebook view as we assume a larger number of attributes for each customer. At this stage, the details of this composite view are not yet specified. Semantic relations between different object classes (e.g. between ‚customer´and 'order') as modelled e.g. in typical domain object models are realized through corresponding navigation paths by normalizing the cardinalities of these relations through appropriate collection views (e.g. by a list of orders given by a particular customer).

Element (a) additionally shows the possibility to access objects through filters (e.g. Customers in Southern Germany) which can be defined by the developer or the user. By this mechanism, the interface can be adapted to recurring tasks and object collections needed for a particular purpose. The complete sub-structure of the search dialogue for customers can be defined as a generic building block and used in the same way for the class ‚Order´ or other object classes (b). Building blocks are defined in separate diagrams in order to make them reusable for different dialogues or systems. (c) shows a functional navigation path, which is provided in addition to the overall object-oriented navigation used in this example. This function provides a shortcut by means of an icon or a menu entry ‚New Customer´. Functional navigation paths may be added in order to achieve higher efficiency for repetitive tasks and can be super-imposed on an object-oriented structure which is used as a consistent and flexible basis of the overall system. On the basis of an initial ViewNet representation, dialogue sequences can be further optimized with respect to the users´ tasks which may lead to changing the type and contents of some of the views. In order to represent 'order data' and 'order items', for example, one could argue that a composed master-detail view showing both kinds of data simultaneously will be more transparent to the user and can save dialogue steps (Figure 5). A ViewNet model forms a useful basis for such optimizations, particularly for translating object relations with cardinalities 1:n or m:n into appropriate navigation sequences. The immediate visibility of the view types facilitates the conceptual design and supports the developer in providing dialogues which are adequate to the task.

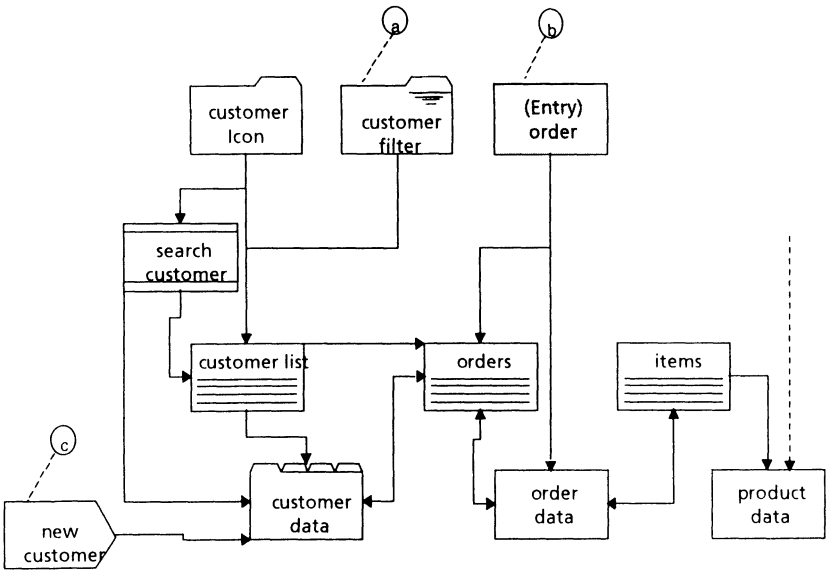

Figure 4: Example of a navigational structure modelled by a ViewNet (order management system).

ViewNets are not only useful for applications of the class-instance type in which more or less fixed views (e.g. screen forms) are filled with changing contents. They can also represent navigation between hypertext-like information units or mixed forms.

Figure 6 shows a part of an internet-based product information system which utilizes both database and hypertext components. If an appropriate clustering is used for the hypertext part of the system, a good overview representation can be maintained. The designer can quickly distinguish between the hypertext- and database-oriented parts of the system and optimize it according to the users'needs.

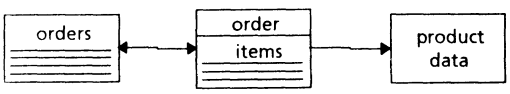

Figure 5: Optimized views and navigation steps for a part of the example in figure 4.

5 DISCUSSION

The ViewNet modelling technique presented here facilitates an understanding of the role and semantics of the different views involved in the navigation structure of a system. This contrasts with existing dialogue modelling techniques which focus on describing the dynamics of the dialogue in detail. The development of the conceptual structure of an application is supported by introducing different types of views. By using graphic pictograms to represent the different types of views, the designer as well as the users involved in the development process can more easily understand and modify the navigation structure. It is particularly important that the designers' attention is directed to the principles underlying the design of the navigation structure such as object-oriented versus function-oriented navigation. The trade-offs between different approaches can be highlighted and usability issues arising from the design of the navigation structure be reflected in a systematic manner..

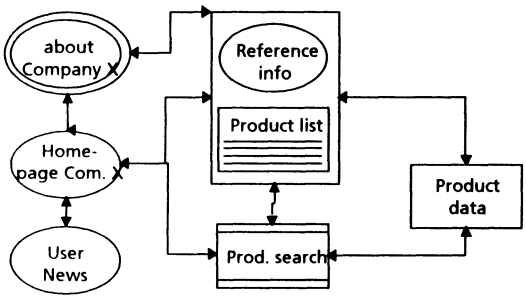

Figure 6: Example of a mixed hypertext/database-oriented navigation.

When applying the ViewNet approach, the problem of structuring the navigation for a large system can be more decomposed into several design steps. In many cases, it will be advantageous to develop first an object-oriented navigation structure as the basis of the application. Additional task- or process-related navigation paths can then be super-imposed on that structure in a second step.

The ViewNet technique can be consistently embedded in an overall software engineering process, especially in object-oriented development methods such as OMT (Object Modelling Technique, Rumbaugh et al. 1991). ViewNet forms part of a larger system development method which comprises the following steps and techniques and which is described in detail in (Ziegler 1997):

- Development of an object model of the application in OMT notation
- Specification of tasks and work processes using a statechart-like representation (Task-Object Charts, Ziegler 1997)
- Systematic, rule-driven derivation of the navigation structure from the object and task model using the ViewNet representation
- Specification of the dynamic behaviour of the dialogue by extending ViewNets into Petri-net based Dialogue Nets
- Visual design of the user interface
-

An initial ViewNet model can be easily extended by including a definition of the dynamics of the dialogue. This can be done by adding transition information, for instance, with the constructs used in dialogue nets. The further refinement of a ViewNet model, however, is beyond the scope of this paper. We claim that, especially in the early design phases, the development of the conceptual structure of the navigation is more important than the dynamic aspects of the interaction.

The ViewNet technique has as yet been applied in a number of development projects and assessed in a qualititative fashion. Its main value was seen in its informality and the intuitive graphical representation which was comprehensible for users participating in the development. The representation technique proved to be particularly useful in group discussions in which the requirements of an application and the initial dialogue strucures were developed. In such situations it is important that all participants can see the design on a large whiteboard or display. Currently, the technique is applied in a paper-based form or by using of a standard diagramming tool with predefined ViewNet symbols. Future work will focus on appropriate tool support, particularly for working in joint development sessions.

6. REFERENCES

Berk, E. & Devlin, J. (1991): Hypertext/Hypermedia Handbook. New York: McGraw-Hill.

Denert, E. (1977): Specification and design of dialog system with state transition diagrams. In Morlet, E. & Ribbens, D. (Eds.), Proc. Int. Computing Symposium. Amsterdam: Noth-Holland, 417-427.

Jacob, R.J.K. (1986): A specification language for direct manipulation user interfaces. ACM Transactions on Graphics, Vol. 5, 283-317.

Janssen, C.; Weisbecker, A. & Ziegler, J. (1993): Generating user interfaces from data models and dialogue net specifications. In Proceedings of INTERCHI (Amsterdam, 24-29 April), New York: ACM, 418-423.

Janssen, Chr. (1993): Dialognetze zur Beschreibung von Dialogabläufen in graphisch-interaktiven Systemen. In K.-H. Rödiger (Hrsg.): Software-Ergonomie '93. Stuttgart: Teubner, 67-76 (in German).

Nielsen, J. (1990): The art of navigating through hypertext. Communications of the ACM, March 1990, Vol. 33, No. 3, 296-310

Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F. & Lorensen, W. (1991): Object-Oriented Modelling and Design. Englewood Cliffs N.J.: Prentice-Hall.

Ziegler, J. E. (1997): Eine Vorgehensweise zur objektorientierten Entwicklung graphisch-interaktiver Informationssysteme. Heidelberg: Springer-Verlag (in German).