Graph Access Pattern Diagrams (GAP-D): Towards a Unified Approach for Modeling Navigation over Hierarchical, Linear and Networked Structures

Matthias Keller and Martin Nussbaumer

Steinbuch Centre for Computing (SCC)
Karlsruhe Institute of Technology (KIT) D-76128 Karlsruhe, Germany
{Matthias.keller,martin.nussbaumer}@kit.edu

Abstract. In this paper we motivate the advantages of a unified, language-independent concept for describing and defining navigation systems based on underlying graph structures. We expect that such an approach will lower the effort for implementing navigation systems with application frameworks while increasing the configurability and reusability of navigation systems at the same time. It also allows adapting navigation components to new data sources easily. A visual notation called Graph Access Pattern Diagrams (GAP-Ds) is outlined and its expressivity is demonstrated by examples.

Keywords: Navigation Systems, Graphs, Content Organization.

1 Introduction

The non-scientific standard works on navigation design and Web information architecture (e.g. [1],[2] or [3]) distinguish between different basic types of content organization systems on the one hand and navigation systems build on top of it on the other hand. Organization systems define *relations* on content *items* on an abstract level. In contrast navigation systems provide *hyperlinks* between *pages* representing content items. The hyperlink structure differs from the content structure. In a hierarchical content structure e.g. the first level nodes and the third level nodes are not connected directly, but there will be a hyperlink from all third level pages to all first level pages in a global navigation system based on that hierarchy.

If the underlying organization system is modeled as graph, different navigation systems can be described by simple patterns (**Fig. 1**). For example a local navigation may render the hyperlinks to the page representing the parent item in the organization system and to all the children. The idea presented in this paper is to develop a formal notation for this kind of pattern. Our vision is a general, lightweight, application-independent concept for defining common navigation systems such as menus or navigation aids that are based on hierarchical, linear or networked content structures which can be modeled as graphs. According to [1] other forms of content organization are the database-model and social classification. The Web Engineering methods as WebML [4] or OOHDM [5] focus on the database-content-model and provide elaborated models for this purpose.

We experienced that although the largest part of common navigation systems belong to one of a few types such as global, local, supplementary or courtesy navigation [2], there are a plethora of ways how navigation systems translate graphs into hyperlink structures. A navigation system implementing a hierarchy e.g. may always expand all levels or just the active level. Parent levels may be expanded or closed. Children may be visible permanently or only when the mouse pointer is moved over the parent element, etc.

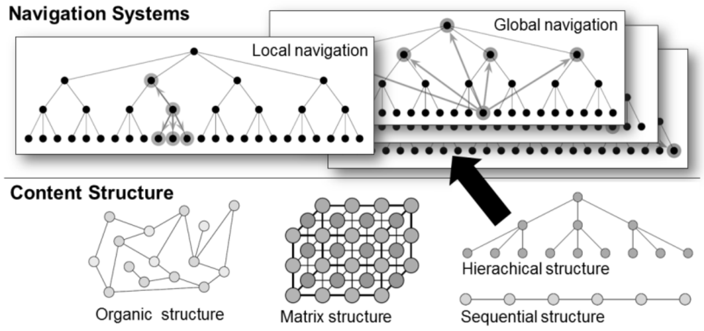

Fig. 1. Navigation systems as access patterns on top of graphs. Illustrations taken from [2].

We want to transform the intuitive concept of classifying navigation systems based on graphs into a small, simple and highly focused language. According to the idea of Domain-Specific Languages (DSLs) a Domain Interaction Model (DIM) as intuitive graphical notation is outlined in this paper [6].

Web application frameworks such as content management systems usually include components implementing the basic types of Web navigation on top of structures that can be modeled as graphs, e.g. folder-like structures (hierarchical graphs) or lists. A problem is that the navigation components provided, their behavior, the extent of configurability and the way they can be configured depends on the application.

2 Benefits

The proposed approach separates three common layers: The data layer, the navigation layer and the presentation layer. The data layer provides an abstraction for modeling the content organization as a graph with two types of relations. The *navigation layer* defines the hyperlink structure with GAP-Ds that is rendered by a processor e.g. in the form of nested lists. The presentation layer consists of presentation templates that guarantee a consistent visualization of all possible results.

We expect benefits for all three layers: On the data layer the graph model is an abstraction that allows adapting new data sources more easily. On the navigation layer GAP-Ds complement the concept of HCI interaction patterns [7] by a formal method for describing the behavior of navigation systems in detail. The current lack of such a method may be a reason for the low consensus on identified interaction patterns compared to software engineering design patterns that can be described by UML. GAP-Ds allow specifying navigation systems in corporate design style guides precisely and extending the configurability of navigation systems in application frameworks. GAP-Ds reduce the effort of implementing navigation systems. GAP-Ds can be reused and shared beyond the scope an application or framework. Considering GAP-Ds also makes presentation templates more universal and allows combining them with a broader range of navigation systems.

3 Outline of a Notation for GAP-Ds

In this section we outline the basic elements (**Fig. 2**) of a notation for GAP-Ds that we are discussing at the moment and demonstrate their expressivity by examples. Since in the proposed approach navigation systems are modeled with GAP-Ds on top of graphs, a notation for describing them is the foundation.

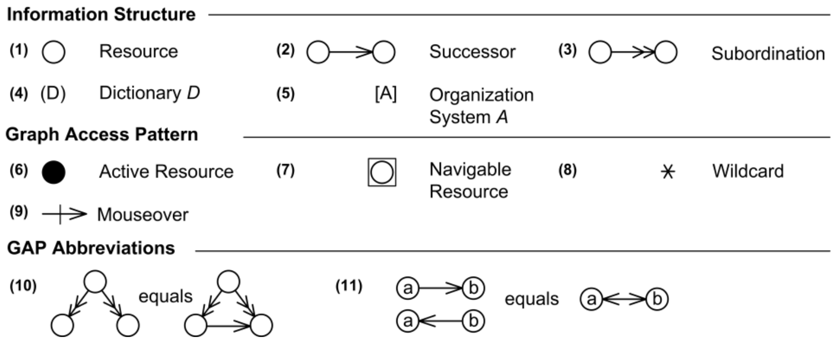

Fig. 2. GAP-D notation

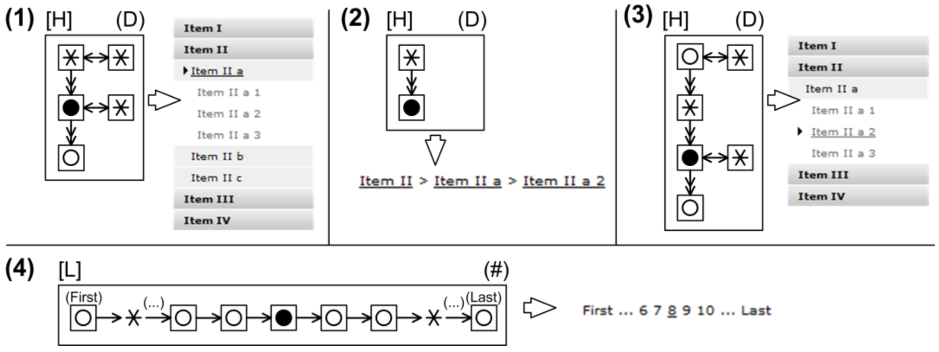

Fig. 3. GAP-D examples for a hierarchical organization system [H] and a linear organization system [L]: (1) All levels are expanded; (2) Active reference / breadcrumb; (3) First and current level and all ancestors and children are expanded (4) Paging

We propose to model organization systems as a graph with resources (1) as nodes and two types of edges, one representing succession (2) and the other representing subordination (3). When modeling hierarchies with the subordination relation, siblings are always considered as successors too (10). A graph representing an organization system can be named and referred by [*name*] (5). The model also contains dictionaries that associate names with resources (4). This allows using different labels for the same resource depending on the navigation system. GAP-Ds describe the navigation options depending on the active resource (6). The content relations are used to select neighbored resources. E.g. with the subordination symbol (3) all children of the active resource can be selected. A square indicates that that a hyperlink to a resource is displayed (7). A wildcard symbol models repetition (8). A line crossing a relation symbol (9) indicates a mouseover-effect. Finally an abbreviation for modeling bi-directional relations seems useful (11). The examples in **Fig. 3** illustrate the expressivity of these few elements.

4 Discussion and Ongoing Work

Before developing tools, we want to include the feedback from the community in our considerations. We are also planning to analyze selected Web sites in order to evaluate which percentage of navigation systems can be described with GAP-Ds and how the notation can be extended while keeping the balance between simplicity and expressivity. It should be evaluated how quick developers can adopt and use the concept. Implementing a GAP-D editor and developing application-specific Solution Building Blocks [6] that are able to process GAP-Ds would be the next steps.

References

1. Morville, P.: Information architecture for the World Wide Web. O'Reilly, Sebastopol (2007)
2. Garrett, J.: The elements of user experience: user-centered design for the web. American Institute of Graphic Arts, New Riders (2003)
3. Kalbach, J.: Designing Web navigation. O'Reilly, Beijing (2007)
4. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann Publishers Inc. (2002)
5. Schwabe, D., Rossi, G., Barbosa, S.D.J.: Systematic hypermedia application design with OOHDM. In: Proceedings of the the Seventh ACM Conference on Hypertext, pp. 116–128. ACM, Bethesda (1996)
6. Nussbaumer, M., Freudenstein, P., Gaedke, M.: The Impact of Domain-Specific Languages for Assembling Web Applications. The Journal Engineering Letters 13, 387–396 (2006)
7. Kruschitz, C., Hitz, M.: Analyzing the HCI Design Pattern Variety. In: Proceedings of AsianPLoP 2010 (2010)