

# **Adaptive Layout of Dynamic Web Pages**

Alexander Kröner

**EXHIBIT**

2007

# Adaptive Layout of Dynamic Web Pages

Alexander Kröner



Alexander Kröner  
Deutsches Forschungszentrum für Künstliche Intelligenz GmbH  
Stuhlsatzenhausweg 3  
66123 Saarbrücken  
Deutschland  
E-mail: kroener@dfki.de

Dissertation  
zur Erlangung des Grades  
Doktor der Ingenieurwissenschaften (Dr.-Ing.)  
der Naturwissenschaftlich-Technischen Fakultät I  
der Universität des Saarlandes

|               |                                        |
|---------------|----------------------------------------|
| Dekan:        | Prof. Dr. Rainer Schulze-Pillot-Ziemen |
| Vorsitzender: | Prof. Dr. Philipp Slusallek            |
| Gutachter:    | Prof. Dr. Wolfgang Wahlster            |
|               | Prof. Dr. R.G. (Randy) Goebel          |

Tag des Kolloquiums: 21. Dezember 2000

**Kröner, Alexander:**  
Adaptive Layout of Dynamic Web Pages /  
Alexander Kröner. – Berlin : Akad. Verl.-Ges. Aka;  
Amsterdam : IOS Press, 2001  
(Dissertationen zur künstlichen Intelligenz ; Bd. 248)  
Zugl.: Saarbrücken, Univ., Diss., 2001  
ISBN 3-89838-248-6  
ISBN 1-58603-189-9

© 2001 Akademische Verlagsgesellschaft Aka GmbH, Berlin  
Unrevised second printing, 2003

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior permission from the publisher.

“infix” is a joint imprint of Akademische Verlagsgesellschaft Aka GmbH (Berlin) and IOS Press BV (Amsterdam)

Reproduced from PDF supplied by the author  
Printing and Binding: Hundt Druck GmbH, Köln  
Printed in Germany

ISSN 0941-5769  
ISBN 3-89838-248-6  
ISBN 1-58603-189-9

# Contents

|          |                                            |           |
|----------|--------------------------------------------|-----------|
| <b>1</b> | <b><u>Introduction</u></b>                 | <b>1</b>  |
| 1.1      | <u>The World Wide Web</u>                  | 1         |
| 1.2      | <u>Presentation Layout</u>                 | 4         |
| 1.3      | <u>Web Page Layout</u>                     | 6         |
| 1.3.1    | <u>Creation</u>                            | 6         |
| 1.3.2    | <u>Tools</u>                               | 9         |
| 1.4      | <u>About this Work</u>                     | 10        |
| 1.4.1    | <u>Goals</u>                               | 11        |
| 1.4.2    | <u>Framework</u>                           | 11        |
| 1.4.3    | <u>Approach</u>                            | 13        |
| 1.4.4    | <u>Structure</u>                           | 14        |
| <b>2</b> | <b><u>Layout Basics</u></b>                | <b>17</b> |
| 2.1      | <u>Terminology</u>                         | 17        |
| 2.2      | <u>Aspects of Layout</u>                   | 21        |
| 2.2.1    | <u>Physical Aspects of Layout</u>          | 21        |
| 2.2.2    | <u>Communicative Aspects of Layout</u>     | 22        |
| 2.2.3    | <u>Dynamic Layout</u>                      | 24        |
| 2.3      | <u>The Layout Problem</u>                  | 25        |
| 2.3.1    | <u>Nature and Size of the Search Space</u> | 26        |
| 2.4      | <u>Layout Techniques</u>                   | 28        |
| 2.4.1    | <u>Styles</u>                              | 28        |
| 2.4.2    | <u>Scoping</u>                             | 29        |
| 2.4.3    | <u>Placement</u>                           | 30        |
| 2.5      | <u>Approaches</u>                          | 34        |
| 2.5.1    | <u>Measuring Effectiveness</u>             | 34        |
| 2.5.2    | <u>Generate and Test</u>                   | 35        |
| 2.5.3    | <u>Procedural Programming</u>              | 36        |
| 2.5.4    | <u>Rule-based Systems</u>                  | 36        |
| 2.5.5    | <u>Pattern-based Techniques</u>            | 38        |
| 2.5.6    | <u>Constraints</u>                         | 38        |
| 2.6      | <u>Context and Layout</u>                  | 44        |
| 2.7      | <u>State of the Art</u>                    | 47        |

|          |                                                  |           |          |                                         |            |
|----------|--------------------------------------------------|-----------|----------|-----------------------------------------|------------|
| 2.7.1    | SGML . . . . .                                   | 47        | 3.7      | Working with the IOM . . . . .          | 119        |
| 2.7.2    | HTML . . . . .                                   | 48        | 3.7.1    | Connecting IOM Instances . . . . .      | 119        |
| 2.7.3    | Cascading Style Sheets . . . . .                 | 50        | 3.7.2    | Connecting LKA and IOM . . . . .        | 123        |
| 2.7.4    | XML . . . . .                                    | 52        |          |                                         |            |
| 2.7.5    | XSL . . . . .                                    | 54        | <b>4</b> | <b>Layout Knowledge Processing</b>      | <b>127</b> |
| 2.7.6    | Context-based Web Pages . . . . .                | 55        | 4.1      | Basics . . . . .                        | 127        |
| 2.7.7    | Constraint-based Web Layout . . . . .            | 57        | 4.2      | Overview . . . . .                      | 129        |
| 2.7.8    | Automated Presentation Generation: WIP . . . . . | 59        | 4.2.1    | General . . . . .                       | 130        |
| 2.7.9    | Visual Language Parsing . . . . .                | 63        | 4.2.2    | Processing Flow . . . . .               | 130        |
| 2.7.10   | Dynamic Web Pages: RadioWeb . . . . .            | 65        | 4.3      | Pre-Processing . . . . .                | 131        |
| 2.7.11   | Other Applications . . . . .                     | 66        | 4.3.1    | Context Retrieval . . . . .             | 131        |
| 2.7.12   | Summary . . . . .                                | 69        | 4.3.2    | Knowledge Base . . . . .                | 131        |
| 2.8      | Summary . . . . .                                | 70        | 4.3.3    | Document Structure . . . . .            | 132        |
| <b>3</b> | <b>Layout Knowledge Representation</b>           | <b>71</b> | 4.4      | Document Tree . . . . .                 | 133        |
| 3.1      | Layout User Groups . . . . .                     | 72        | 4.4.1    | Interface . . . . .                     | 133        |
| 3.2      | Layout Knowledge Architecture . . . . .          | 73        | 4.4.2    | Processing . . . . .                    | 133        |
| 3.3      | Knowledge Specification . . . . .                | 74        | 4.5      | Design Command Processing . . . . .     | 134        |
| 3.3.1    | Overview . . . . .                               | 75        | 4.5.1    | Interface . . . . .                     | 134        |
| 3.3.2    | Parameters . . . . .                             | 76        | 4.5.2    | Evaluation . . . . .                    | 134        |
| 3.3.3    | Validation and Assignment Constraints . . . . .  | 79        | 4.6      | Design Pattern Processing . . . . .     | 135        |
| 3.3.4    | Format Constraints . . . . .                     | 84        | 4.6.1    | Interface . . . . .                     | 135        |
| 3.3.5    | Composition Rules . . . . .                      | 88        | 4.6.2    | Evaluation . . . . .                    | 135        |
| 3.3.6    | Event Rules . . . . .                            | 89        | 4.7      | Composition Rule Processing . . . . .   | 136        |
| 3.3.7    | Design Patterns . . . . .                        | 91        | 4.7.1    | Interface . . . . .                     | 136        |
| 3.3.8    | Design Commands . . . . .                        | 95        | 4.7.2    | Evaluation . . . . .                    | 136        |
| 3.3.9    | Using Annotations . . . . .                      | 97        | 4.8      | Event Processing . . . . .              | 137        |
| 3.4      | Layout Styles . . . . .                          | 99        | 4.8.1    | Interface . . . . .                     | 137        |
| 3.4.1    | Using and Creating Styles . . . . .              | 99        | 4.8.2    | Evaluation . . . . .                    | 137        |
| 3.4.2    | Merging Styles . . . . .                         | 100       | 4.9      | Format Solving . . . . .                | 140        |
| 3.4.3    | Examples . . . . .                               | 102       | 4.9.1    | Interface . . . . .                     | 140        |
| 3.5      | Miscellaneous . . . . .                          | 105       | 4.9.2    | Evaluation . . . . .                    | 140        |
| 3.5.1    | Context Model . . . . .                          | 105       | 4.9.3    | Working with Format Variables . . . . . | 141        |
| 3.5.2    | Decision Patterns . . . . .                      | 109       | 4.10     | VA Solving . . . . .                    | 143        |
| 3.5.3    | Non-declarative Knowledge . . . . .              | 111       | 4.10.1   | Interface . . . . .                     | 143        |
| 3.6      | Internal Object Model . . . . .                  | 112       | 4.10.2   | Evaluation . . . . .                    | 144        |
| 3.6.1    | Logical Layout Objects . . . . .                 | 114       | 4.11     | Post-Processing . . . . .               | 150        |
| 3.6.2    | Multimedia Objects . . . . .                     | 114       | 4.11.1   | Beautifying Layout . . . . .            | 151        |
| 3.6.3    | Physical Layout Objects . . . . .                | 114       | 4.11.2   | Extracting Decision Patterns . . . . .  | 152        |
| 3.6.4    | Reference Objects . . . . .                      | 116       | 4.11.3   | Output Generation . . . . .             | 153        |
| 3.6.5    | Design Objects . . . . .                         | 117       | 4.12     | Runtime . . . . .                       | 153        |
| 3.6.6    | Logical Container Objects . . . . .              | 117       | 4.12.1   | Pre-Processing . . . . .                | 154        |
| 3.6.7    | Physical Container Objects . . . . .             | 118       | 4.12.2   | Layout Generation . . . . .             | 155        |
| 3.6.8    | Frame Objects . . . . .                          | 118       | 4.12.3   | Post-Processing . . . . .               | 155        |
|          |                                                  |           | 4.12.4   | Summary . . . . .                       | 156        |



# Chapter 1

## Introduction

Today we are the members of an information society in which each member may provide information and everyone has access to nearly every kind of information in huge amounts. As many providers are offering similar information, each provider is challenged to compete for the same set of customers.

To be successful in this competitive environment, several aspects have to be regarded. First, the quality of the product itself is important. Product information, especially when information itself is the product, has to be up to date and informative, and should be presented in an original, entertaining way. A customer-oriented usage and appearance of the presentations conveying the offer should be provided as an additional service to improve acceptance. This makes a presentation easier to use and to understand, and personalizes the contact to the reader.

Another point is a unique provider appearance: since providers typically offer several products, it is important that the consumer's first contact leaves a good impression to increase the probability of his return to check other offers. To achieve this, a provider has to present his offer—and himself—in a manner uniquely different in appearance from that of his competitors. Very important is also the consistency of the appearance: look and feel of a provider's presentation should be consistent with each customer contact.

The following work will discuss problems in the area of multimedia presentations and show ways to solve them, where the focus is on presentations in the World Wide Web.

### 1.1 The World Wide Web

Originally, the idea of the World Wide Web (WWW) was to provide a simple-to-use basis for the exchange of scientific information between research facilities all over the world. At that time, FTP and GOPHER were the most common techniques to exchange such information, but beneath some technical weaknesses, both were characterized by cumbersome usage

as they made little use of the possibilities of graphical user interfaces. Furthermore, the display of the requested information was not (when using FTP) or little supported.

These limitations were fixed in the WWW by the HyperText Markup Language (HTML). This language enabled authors to enrich their documents with annotations which included simple assertions about the content's nature as well as references to other documents anywhere in the WWW. These annotations had to be translated by the display (or rendering) software, usually called *browser*, regardless of the platform on which this browser was implemented. Thus a worldwide hypertext structure was born, and due to its simplicity, in only a few years browsers became available for all major platforms. The result was an explosion of information presented in the WWW (for instance, see the statistics provided by the Internet Society [ISO00] or by CommerceNet [Com99]).

In the years to follow, the original HTML was extended step by step to satisfy the requests of presentation authors for additional formatting elements. As a consequence, HTML became considerably complex so that finally experts were required to handle state-of-the-art presentations. Thus a new profession was born, that of the Web page designer, in principle a combination of layout designer and programmer.

Today, professional Web pages created in HTML are highly complex multimedia presentations, typically interactive and well designed. And these features are required, because the WWW has become the key element of the modern information society. It has left its origin at the universities and is now used not only by most companies, but also by day-to-day people all over the world.

Thus a new scenario has arisen: for companies it is no longer enough just to be found in the Web—their appearance has to meet all the requirements we have already mentioned before. In particular it becomes more and more important to provide not only valuable information but also to present it in a pleasant way which makes a company's Web presence more appealing to potential customers than the offer of competitors.

Making the presented content more valuable for the reader affects not only the quality of the content elements itself, but especially the content selection. It should reflect the reader's interests, optionally enriched by some related information. Since one hopes to have many readers when offering presentations in the Web, there are two ways to realize that task.

An often practiced approach is to merge much information in one presentation, hoping that each individual reader will find what he is looking for. This solution has several disadvantages, where the biggest one is that a reader has to read extensively the visited Web page to find the particular piece of information he is interested in. The second disadvantage is that it is difficult to produce an attractive layout for pages with highly heterogeneous information. Since the space in the browser window is limited, one may

pack the information using small fonts and images—which in turn makes it difficult to read. Alternatively, the page could be made considerably larger than the window—then a certain part of the presentation can only be made visible by scrolling. In both cases it is probable that readers will overlook the requested information, an unsatisfying result for both reader and provider.

Thus more and more providers prefer a second approach to presenting information in the Web: the presentation content is adapted to individual user preferences, a process called personalization. Creating a layout for such presentations is connected with the following considerations.

- **Dynamic content** makes it difficult to create the layout in advance; when the content is extracted from large data bases, it might even be impossible.
- **Numerous requests** are the interest of a content provider, and in the worst case, for each of them a new layout is required. Even if a human designer might cope with this task, it would very cumbersome for him to do a similar work again and again due to changes in the presentation content.
- **Little time between request and response** is available for a designer to do his job, because one cannot expect a reader to be very patient.

Since the regular way of creating a layout by hand is obviously little suited here, one has to think about the automation of this process. To enable the computation of layout, one needs a representation of knowledge about layout. And this leads to the next problem: to create such a representation, tools are required which enable layout designers to express their knowledge in a formal way.

As a consequence, tools for the specification of layout knowledge are of an increasing interest. These tools have to reflect another aspect of the today's Web community: most of its members are inexperienced in layout design, but nevertheless many of them are interested in creating Web pages. Therefore layout tools have to meet the demands of laymen as well as skilled layout designers.

An issue related to these topics is the maintenance of Web presentations. Especially for large companies, it becomes an increasing problem to maintain their often huge Web sites. Here an effective management of content as well as layout information is required to enable fast modification of these presentations. A typical application scenario is a Web-based newspaper where the content is frequently changed, and where on the other side the presentation layout should express the corporate identity by a specific layout. Thus authors need fast access to the content structures only, where layout

designers need tools which enable them to change the corporate layout in just a few actions, and not by revising all affected documents.

Other issues of interest when talking about the present and the future of the Web are the structure and character of content and the growing number of hardware clients (e.g., palmtops, wireless phones, Web TV and many more) which gain access to the Web. Again, these topics affect the representation of both content and layout. For the former, a more sophisticated representation of content structures should be accompanied by a more sophisticated representation of layout related information. For the latter, just imagine a typical Web page designed for a PC with a true-color, large-scale monitor, and a keyboard and a mouse. And then think about how to display and to access this page on a wireless phone with a small grayscale display and the regular phone keys. In the worst case, a completely new presentation design is required, where unnecessary pieces of content are omitted and the layout is reformatted (e.g., uses smaller fonts and pictures). This example demonstrates very well why a powerful and flexible representation of content and layout is required.

To conclude, just saying that there is still some movement in the development of the WWW would little express the high speed with which the Web is changing to a more and more powerful information network that will not only contain nearly each kind of information, but which will also be accessed by nearly everyone in the world for manifold purposes using utilities we currently cannot imagine. Nevertheless one may recognize today two basic tasks which are shared by most of the current trends: the need to specify and to process knowledge about content and layout efficiently.

## 1.2 Presentation Layout

In the previous section we agreed that the questions of how to represent knowledge about layout and how to process it efficiently is important to answer to come to Web-based presentations which are ready for the future requirements. For general multimedia presentations, these questions have been under discussion for several years. We will summarize now some of the basic knowledge gained during research in this area, and then return to the more specific topic of the layout of Web pages.

The general layout problem may be divided into several classes of specific layout problems, some of which are mentioned here.

- **Table layout** is a very specific layout class, especially appropriate for presenting structured information in a way which provides fast access by the reader to the listed information fragments.
- **Graph layout** is another very specific class, often used to visualize relationships between some information units. The relations are dis-

played using lines which connect the related objects. The problem typically consists here in the task of creating a layout for the objects and the lines which makes it easy for the reader to identify which objects are related. Often this problem is constrained by space restrictions and the need to avoid crossing lines. Examples include flow charts, class trees, network diagrams or circuit layout.

- As graphical user interfaces (GUI's) became available, **screen layout** started to be of interest. Basically it is the task of arranging the items shown on a screen (typically windows, buttons and other well-known GUI components) in a way which provides fast access by the user to the elements he is interested in. Here the arrangement should express some kind of importance to reflect the system's ideas about the users needs, and it has to be created in real time, i.e. while the user is interacting with the system.
- The layout of **illustrated documents** is basically the problem of arranging multimedia data such as images, text and movies in such a way that book-like presentations are the result. Common subclasses are regular story books, manuals and newspapers. An interesting aspect of these documents is that they often enclose contents of differing quality, and provide some kind of an access structure which enables the reader to navigate in the document. These navigation structures may be just a table of contents, but also hyperlinks when electronic media are used.
- Including **animations** in presentations makes the layout problem very challenging. For instance, moving elements make it difficult to prevent (or at least to reduce) the unwanted overlapping of layout objects, and time becomes a crucial element, especially when parts of a layout reference other parts—and which have therefore to be visible at the same time.

In the following we will concentrate on the layout of illustrated documents, because this class includes the Web-based presentations we are interested in. Thus when the term *multimedia presentation* is mentioned, we think of an illustrated document. Furthermore, we will use the notion *layout* in a specific way, where temporal or acoustical aspects are relaxed.

The layout of multimedia presentations is determined by a large number of layout aspects of different quality. Primary, physical aspects such as the position and size of the displayed objects enable the transfer of a layout to an output medium. Communicative aspects may be used to emphasize intention and rhetoric of the related presentation. Further aspects arise from the increasing importance of electronic media: here the presentation content—and the layout—may be dynamic, and there may be active elements,

which enable the interaction with a reader beyond the “classic” one-direction information transfer, and which may also affect the presentation layout.

In the practice of layout software development, usually one has to deal with further problems. Layout knowledge provided by human designers is often incomplete or contradictory, which makes complex solving strategies necessary since **a general layout solver still seems out of reach today**. Competing optimization criteria can make it difficult to find an optimal solution for a given problem. A common example for such an optimization problem is the minimization of the free space in combination with time bounds for the generation process. The potential conflict is obvious: if the time bounds are exceeded during the minimization process and there is still free space left, the bounds have to be relaxed, or a sub-optimal result has to be accepted.

### 1.3 Web Page Layout

Web pages are a subclass of multimedia presentations which is characterized by multimedia components, hyperlinks and many other elements. The layout of Web pages includes all the layout aspects mentioned before, and is heavily affected by the process which takes place when a Web page is transferred from an information provider to a reader. We will now have a closer look at this process, and then discuss some of its crucial aspects.

#### 1.3.1 Creation

Figure 1.1 illustrates how the layout of a Web page is produced.<sup>1</sup> Basically, the layout production process can be divided in four phases:

1. **Composing:** Here the HTML or XML source which is the basis of a Web page is assembled.<sup>2</sup> This process takes place on the server side, and is carried out manually, semi-automated or fully automated.
- Authoring by hand is common when creating static parts of a Web page; automation enables the integration of dynamic content in a Web page. Automation means here typically the application of HTML templates using CGI scripts or Java Servlets.
- The composition affects the presentation layout indirectly: during authoring of HTML and HTML templates, authors try to anticipate the rules governing HTML layout which are applied in the rendering step.

<sup>1</sup>This is only a sample illustration: the many tools involved in this process may be combined in several ways. And do further note that there are many Web pages which rely only on a subset of these tools.

<sup>2</sup>A more detailed description of HTML, XML and other representation languages is given in section 2.7.

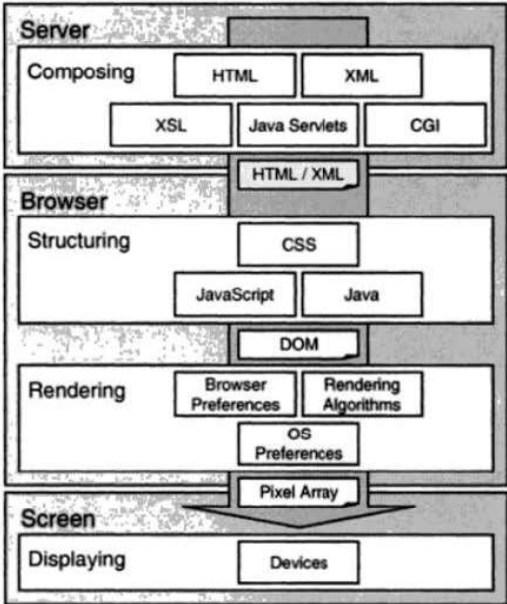


Figure 1.1: The Web layout creation pipeline

2. **Structuring:** When the HTML or XML source is available, it is sent to the reader’s browser. A structuring of the presentation follows, where further details are added which describe the layout in more detail. In particular, style sheets (also provided by the content provider) are applied, and Java as well as JavaScript may affect the presentation. The result of this phase is a DOM [ABC<sup>+</sup>98] instance.
3. **Rendering:** Here layout instructions which are encoded in the presentation are visualized. This process is based on rules governing the visualization of HTML content, but is also heavily affected by the algorithms applied by the browsing software as well as user preferences provided by the browser and the operating system.
4. **Displaying:** Finally, the result of the rendering process, a pixel array,



Figure 1.2: A Web page with static layout

is drawn on the screen. Here device drivers and hardware properties may have some impact on the final layout.

The need for personalized content of Web pages leads to an increased complexity of the composition step: content from various sources has to be dynamically compiled. Therefore a designer has to encode layout knowledge in templates or style sheets which can be applied dynamically just like the content.

The application of this knowledge occurs during structuring and thus is not under the layout designer's control. Since the dynamic content complicates the anticipation of the final layout during the knowledge specification and the current tools for Web layout do not provide mechanisms which ensure the layout's quality, layout conflicts may occur which have their origin in the composition phase.

Another major problem in the layout creation pipeline is the rendering. With so many different kinds of hardware and software accessing the Internet, it is little probable that the result of the rendering will have the same appearance on all these platforms. Thus it is quite difficult for designers to provide a good layout for all these platforms, and consequently it is no wonder that a Web page's layout is often not optimal or even in conflict with the presentation platform.

**Example.** Figure 1.2 shows two screenshots taken from a static Web page. When the browser window is small, much scrolling is needed to reach the invisible content—here a layout with less images, but more text would be helpful. On the contrary, when the browser window is very large, space is left unused on the right side where additional material could be placed.<sup>a</sup>

<sup>a</sup>How the layout of a similar Web page could be adapted to the browser window is shown in Figure E.3.

An often practiced approach to resolve this problem is to provide several HTML documents for a single content, which are optimized for common combinations of hardware and software. But the increasing number of hardware platforms with Internet access makes this approach little feasible because of its high maintenance costs. Furthermore, Web pages with a dynamic content will usually not match a static layout selected from a set of static layouts. Using templates cannot resolve this conflict completely, because then the templates have to be specified according to the presentation platform—and thus the conflict is only moved to another level.

Finally, an aspect which is only little regarded in the current Web page layout is the integration of reader profiles. Currently, when a reader wants to change a layout he is restricted to the interface of his operating system and her browsing software. Here some basic aspects of the layout like font size or face may be changed, but not more complex elements like the appearance of a navigation structure.

1.3.2 Tools

The first generation of Web pages relied on the primitive HTML 1.0 only, and thus simple text editing tools were a popular authoring tool. But the increasing complexity of HTML, the introduction of JavaScript, style sheets and other more sophisticated techniques made this simple authoring approach soon unsuited for authors with less training in the latest techniques. And since there was and is an increasing amount of such authors, more sophisticated toolkits for Web page authoring (e.g., Microsoft Frontpage™) were developed.

The GUI's of such authoring software provide an abstraction from the script level and encourage even laymen to create their own presentations. On the technical level, these tools address the HTML assembly phase. Typically they rely on HTML templates and thus share the disadvantages of the current layout production process. Furthermore, the provided templates are designed very general, and do often include a large overhead of unnecessary HTML code. The resulting Web pages waste bandwidth and share a certain appearance—both reasons which force professional Web page designers to do some post-editing or to refrain from using these tools.

Thus designers still have to cope with the sometimes cryptic HTML code, which makes it often quite difficult to identify which particular task a certain fragment of HTML code has. For both layout designers and authors this is an unsatisfying situation, because it makes it harder for them to maintain their part of a document.

**Example.** The following code is a mix of HTML and JavaScript as found in many Web pages. Information about layout is spread over the whole document: table-cells are used for positioning, font assignments, and blank images to provide spacing.

```
<HTML>
<BODY BGCOLOR="#FFFFFF"><FONT COLOR="#FFFFFF">
  <TABLE BORDER=0 CELSPACING=0 CELLPADDING=0>
    <TR><TD><NOBR>
      <FONT FACE="Arial,Helvetica" SIZE=1>
        <A HREF="http://www.somewhere.com/" TARGET="stage"
          onMouseOver="parent.stat('Department of
            Science in Somewhere');return true">
          <IMG ALIGN=LEFT HSPACE=2 BORDER=0
            SRC="..pic/open_fol.gif">
          <B>Department of Science</B>
        </A>
        <IMG ALIGN=top SRC="..pic/space.gif">
      </FONT>
    </TD></TR>
    <TR><TD><NOBR>
      <FONT FACE="Arial,Helvetica" SIZE=1>
        <IMG ALIGN=LEFT HSPACE=2 SRC="..pic/blank2.gif">
      </FONT>
    </TD></TR>
  </TABLE>
</BODY>
...

```

Beneath this problem, these authoring tools provide only little support for the specification of a layout for Web pages with dynamic content. Mechanisms which facilitate the anticipation of the layout on various presentation platforms are missing as well, and thus these tools cannot be the answer to the problems of the layout creation pipeline we have discussed before.

## 1.4 About this Work

In the following we will summarize the goals of our work, describe our approach to reach these goals, and then conclude with an overview of the structure of this document.

### 1.4.1 Goals

We have elaborated in the last section that the current way of producing a layout for Web pages includes some critical aspects, and thus has the potential to be improved in several ways. This work concentrates on an approach to preventing layout conflicts during composition and rendering. In particular, we want to address the following questions.

1. How can a layout designer be enabled to anticipate the appearance of Web pages with dynamic content, so that he can prevent layout conflicts?
2. How can the prevention and fixing of layout conflicts be automated, so that a designer has not necessarily to be directly involved in the layout creation process?
3. How can information about the presentation environment be integrated into the layout creation process, so that conflicts between layout and environment can be prevented?
4. How can the layout of dynamic Web pages be produced efficiently enough to meet the narrow time constraints of an online scenario?

To answer these questions, we propose an architecture for the automated creation of Web page layout. Based on a separation of knowledge about content and layout, it uses a declarative representation of layout design knowledge and AI-driven processing modules. We show how the proposed architecture provides a mechanism for the efficient specification and processing of layout knowledge for the production of dynamic, user-adapted presentations.

### 1.4.2 Framework

The background of our architecture proposal is the framework shown in Figure 1.3. It contains all components and actions we think to be relevant for a typical access to a dynamic Web page.

#### Processing

In this framework, a typical process takes place as follows. The process begins with a request from a presentation environment (1), which is processed by a presentation server (2). The server's task is to reply to the request with one or more Web pages. Assuming that there is no static Web page available which meets the given request, a Web page has to be created dynamically. Thus a request is sent to the provider data source, which has to provide the document

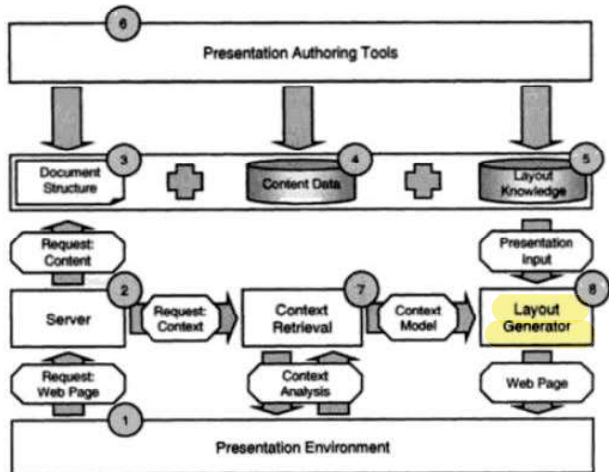


Figure 1.3: A framework for a Web page access

structure (3), the required content data (4) and some layout knowledge (5). These data, typically produced by some authoring tools (6), are forwarded to the layout generator (8).

A further request is sent from the server to a context retrieval module (7). It analyzes the presentation environment and creates a context model which is forwarded to the layout generator (8).

When all required data are available, the layout generator begins to work. Its result is a Web page which is sent back to the presentation environment (1).

Focus

In this work, we address the components of the Web page access framework in the following way.

**Presentation environment (1):** Our discussion of the presentation environment is limited to an analysis which forms the background for our discussion of the notion of “context”. Thus we will neither elaborate how the rendering algorithms applied by common browsing software work, nor how the displaying of the final result is done.

**Server (2):** The layout Web server is basically a wrapper that enables access from the Web to the services provided by the layout generator. It is not discussed in this work.

**Document structure (3):** We propose an XML-based representation of the document structure, and show how this representation may be mapped to layout knowledge.

**Content data (4):** The representation of content data is discussed in this work on the implementation level only.

**Layout knowledge (5):** The representation of layout knowledge is a major topic of this work, and is therefore intensively discussed.

**Authoring components (6):** Andreas Weiss developed a graphical authoring component which enables the editing of the document structure and the layout knowledge base (see Appendix D). Content editing is not discussed here.

**Context retrieval (7):** This component, and the related representation of context data, were originally developed by Markus Kaiser. In this work, we focus on the impact of the presentation environment on layout, and show how context knowledge may be used to adapt layout.

**Layout generator (8):** The layout generator is also a major topic of this work. We will propose an architecture for processing layout knowledge, and describe a sample implementation.

1.4.3 Approach

The foundation of the proposed architecture is a representation which includes knowledge about the presentation environment, layout design, and layout results.

As mentioned before, knowledge about the presentation environment is automatically collected by a context retrieval module in a context model. This model may be referred when knowledge about layout design is specified.

The formal representation of knowledge about layout design was one of the major parts of this work. Thus we developed a declarative language which relies on the following design criteria to provide an effective representation (see also [Be85]).

- **Expressive adequacy** is needed to cover all layout aspects of dynamic Web pages. Thus our language has to enable the specification of “classical” layout aspects (e.g., the spatial arrangement and choice of the font format) as well as layout alternatives, solving strategies and information about the presentation context.

- **Efficient specification** of layout knowledge is required to reduce the efforts required for managing layout knowledge.
- **Reasoning efficiency** is required to compute a layout based on the language's implications in a timely fashion.

The inference process transforms a given document structure in a layout result using the knowledge encoded in our meta-language. The transformation is performed by a so-called *layout generator*, which is founded on the following design criteria:

- **Modularization** enables not only the efficient processing of our meta-language, but also the adaptation of the layout generator to changing processing requirements. Furthermore, clearly defined interfaces enable external software to analyze and to manipulate the layout generation process.
- **Automation** includes the detection of layout faults, the semi or fully automated search for alternative layouts, and a self-optimization of the processor.

Our proposal is accompanied by a sample implementation. To take into account the efforts of organizations like the World Wide Web Consortium for a more standardized representation and processing of multimedia documents, we tried to make use of these emerging standards as far as possible. Thus our work is not only relevant for state-of-the-art software development in this area, but we hope further that it will make it easier to adapt our implementation to future requirements, resulting in an increased acceptance of our approach.

#### 1.4.4 Structure

**Chapter 2 (Basics):** Following the introduction, we describe the basics of our work. We begin with a characterization of the layout problem, followed by a description of techniques used by human layout designers to solve this problem. Then we discuss approaches for the computation of layout, and do elaborate the presentation context's meaning for the creation of layout. A review of state-of-the-art software for the generation of layout completes the basics of this work. Here the focus is on Web-related implementations, but other applications which are of special interest for this topic are taken into account as well.

**Chapter 3 (Representation):** After building a basis for further discussions, we discuss a formal representation of layout knowledge that enables an efficient specification and inference. First the results of our previous analysis are used to find a heuristic model which maps

layout knowledge to various representation techniques. Based on this mapping, we show how to address layout knowledge in a declarative manner using a so-called *layout knowledge representation architecture*. Included is a discussion of how to address knowledge about the presentation context using this architecture, and how self-optimization may improve the efficiency of the layout creation. The chapter ends with a description of the so-called *internal object model*, a representation for layout results.

**Chapter 4 (Processing):** We proceed with a discussion of a modular architecture for the processing of layout knowledge. It includes auxiliary modules (e.g., a pre- and a post-processor) as well as constraint solvers, rule-based techniques and specific processing modules for the abstract layout knowledge elaborated before. For each module the interface is described to improve the understanding of its role in the overall architecture, and an abstract analysis of its mechanics is given.

**Chapter 5 (Implementation):** We have implemented the components in our proposed framework with our *DesignComposer*, a layout generator for Web pages. In this chapter, we discuss some of the most important classes of this Java implementation. The discussion is still abstract not only to simplify the understanding of the processing, but also to simplify the transfer of our ideas to new projects.

**Chapter 6 (Conclusion):** We conclude our proposal with a brief summary of all the work done, and a discussion of its meaning for other research, not only in the domain of automated layout creation, but also of knowledge-based processing. Finally, we look into the future of our approach and its implementation, where possible extensions are discussed which could help to solve problems yet unsolved and to increase the flexibility and efficiency of our approach.

In the appendix of this work we provide additional examples, an excerpt of the manual of our meta-language for the representation of layout knowledge, a description of the DesignComposer demonstrator and some application scenarios. Please note that this part represents only a snapshot of our research which was still under progress when this document was written. Thus we recommend a visit to our homepage (see [Kr99b]) to obtain up-to-date information about our project.