A Model-Driven Approach for the Automatic Display Layout of Web Applications

Davide Mazza
Dipartimento di Elettronica e Informazione
Politecnico di Milano
I-20133 Milano, Italy
Email: mazza@elet.polimi.it

Abstract—This paper presents a model-driven approach for the automatic computation of the layout of Web application pages, that exploits the same conceptual models used in the design phase. Developers can use the same integrated environment to specify the requirements and design of their systems, generate the complete executable source code, and perform the automated layout creation of the application pages through the employment of reusable models. The paper describes the transformations that are at the basis of the layout computation and discusses their implementation.

Index Terms—Model-Driven Engineering, Web Engineering, Data-intensive Web Applications, Modeling languages.

I. INTRODUCTION

Automatic Display Layout is the problem of the definition of a procedure for automatically positioning the contents of a diagram or of a document. Solutions for the layout of nodes and links of graphs, for the positioning of electronic components over boards, or for the layout of text, figures, tables, etc. in document presentations can be found in literature [1]. In this paper the Automatic Display Layout of Web pages specified with WebML [2] is considered. WebML is a visual domain-specific language (DSL) for representing interactive systems, which exploits the Entity/Relationship data model for the specification of the data schema underlying the application, and a specific notation, named hypertext model, to express the structure of the application front-end and user interaction.

WebML along with the related CASE tool [3] represent an example of model-driven development. According to the OMG Model Driven Architecture specifications [4], several Web engineering methodologies support the automatic transformation from PIMs\(^1\) to PSMs\(^2\), like for example [5], [6], where application code is automatically generated from the conceptual models. The automatic display layout procedure described in this paper is instead a PIM-to-PIM transformation: starting from the data and hypertext PIM models, the layout of the presentation model is automatically derived. It consists in a transformation of the two models expressed by a set of rules, grouped in different classes of applicability [7]. Such rules make use of the concepts commonly adopted in the modeling of Web applications, and are reusable for the major part of them. This paper describes in detail the transformation rules and discusses their implementation in the case of WebML as modeling language. It is organized as follows: Section II introduces a running example considered throughout this paper; Section III presents the main concepts of the WebML language; Section IV presents the Automatic Display Layout procedure by focusing on the models employed for rules’ definition and their reusability feature; Section V discusses the implementation of the actual transformation procedure; finally, Section VI presents related work and draws the conclusions.

II. RUNNING EXAMPLE

The running case considered throughout the paper is the Alta Scuola Politecnica (ASP) Web Community application [8]. ASP is a School at Politecnico di Milano and Torino offering an advanced educational programme to selected M.Sc. students; such programme includes innovative courses and multidisciplinary projects and is supported by a Web application, composed of three parts, each addressing a particular class of users: (1) a student area, reserved to ASP students, which gives the possibility to get all the information needed on the courses (locations, calendars, etc.), retrieve the downloadable didactic material of the courses and upload papers (for course evaluation); (2) a teacher area, reserved to courses’ teachers, used to upload didactic material; (3) an administrator area, reserved to the administrators of the Community, which allows the creation, publication and deletion of materials, the insertion of courses’ organization information, and any other operation needed for application management.

III. WEBML IN A NUTSHELL

This section reviews the main concepts of WebML that are needed to understand the proposed layout procedure. For further details on the WebML model the reader may refer to [9]. WebML specifications consist of three models: the data, the hypertext and the presentation model.

Data model. The goal of the data model is enabling the specification of the data used by the application. WebML exploits the most successful and popular notation, namely the Entity-Relationship (E-R) model. Figure 1 shows a little fragment of the WebML data model of the running example. The entity Student contains a fragment of the information about the relevant concepts of a Student of the School. The application supports the students’ subscription (managed by

1Platform-Independent Models.
2Platform-Specific Models.
the entity *Event Registration*) to School events (e.g., teaching sessions), stored in the *Event* entity. Events are offered to all the students, but each student can register to a specific event only once. The Static Contents entity works as a repository for chunks of static text that have to be shown in pages (such as introductions, instructions on how to complete a form, etc.). This is a service entity, not needed for the strict modeling of the problem domain.

![Fig. 1. A fragment of the data model of the running example](image)

**Hypertext model.** The goal of the WebML hypertext model is the specification of the organization of the front-end interfaces of a Web application in terms of pages contents and links. The key ingredients of WebML are: pages, content and operation units for delivering content or performing operations, and links. The content displayed in a unit typically comes from an entity of the data model, and can be determined by means of a condition selector, which is a logical condition to filter the entity instances to be published. Units are the building blocks of pages, which are the actual interface elements delivered to the users. Pages are typically built by assembling several units of various kinds, to attain the desired communication effect. Pages and units do not stand alone, but are connected by links to form a hypertext structure. Links express the possibility of navigation from one point to another in the hypertext, and allow parameters passing among units for the proper computation of the contents to be shown in a page. WebML also allows specifying operations implementing arbitrary business logic; in particular, a set of data update operations is predefined, whereby one can create/delete/modify the instances of an entity, and create or delete the instances of a relationship. As an example, Figure 2 shows a fragment of hypertext. The left-most unit is an index unit defined onto the *Event* entity: it shows an indexed list of events to the user. The right-most unit is a data unit (also defined onto the same entity). Here it is used to show the details about the event selected from the index: the link that connects the two units carries the parameters provided by the source unit and used for computation by the destination unit and is defined as a navigational link. In particular, in the example the link carries the selected event ID, which is then used by the data unit to retrieve only the data of the selected event.

![Fig. 2. A simple page specification using the WebML hypertext model.](image)

**Presentation Model.** The presentation model design in WebML is mainly split into two phases: the definition of the style and the specification of the layout of pages’ contents. First, style guidelines are specified by the definition of: 1) a page grid, i.e., a table containing a specific arrangement of rows, columns, and cells where the contents must be positioned; 2) the content positioning specification, addressing the rules for assigning standard content elements, like banners, menus and submenus; 3) the graphical guidelines, containing the formatting rules for graphical items, like fonts, colors, borders, and margins. They can be expressed by means of Cascading Style Sheets or equivalent specifications. In a second phase, and in particular once the data and the hypertext model have been defined, the layout of the content units of each WebML page have to be specified. In the most common data-intensive Web applications the main content of the pages is usually placed in a central grid, where contents are arranged in multiple rows or columns; further lateral areas on the left and on the right of the grid may be present (as shown on the left-hand side in Figure 3). Currently, in the WebML methodology (and in the commercial tool supporting WebML [3], too) the mapping of the contents specified in the hypertext model to the layout in the presentation model is performed manually by the Web application designer, who positions each unit of each WebML page inside the content grid. It is important to remark here how the WebML language is actually a truly PIM due to the independence of its compound models from any particular implementation technology. The problem addressed by this paper consists in the definition of a procedure that allows to obtain the presentation model of a page given its data and hypertext model, thus defining an actual PIM-to-PIM transformation.

**IV. Automatic Display Layout Procedure**

Given the data and the hypertext models specifying a Web application, its final layout is obtained by a series of subsequent model transformations. For the specification of the transformations we have exploited the visual notation of the conceptual models, in particular, the hypertext and layout models. An immediate mapping from such models to graphs is possible, and the transformations can be expressed in terms of graphs: in the E/R model entities are the nodes and relationships the arcs; in the hypertext model the units of a page are the nodes and links connecting them are the arcs; finally, also the presentation model can be represented as a graph, as shown in Figure 3. In particular, the grid area is a set of rows and columns (the tabular area) in which contents are placed. For each cell of the grid, only one content is allowed. The left and right columns are instead considered as unique (vertical) sequences of contents.

The remarkable feature of the procedure is the possibility to define a series of template schemas for rules’ application conditions and results. These templates leverage on the identification of functional patterns, roles and relationships commonly employed and found in the development of Web applications. These pre-defined models can be stored in repositories, can be exploited as is or edited to tune them to specific needs. The key factor is that they can be reused in a great number of projects, due to their wide application.

**A. Determination of the visualization set**

The layout of pages consists of a set of contents to show. In WebML not all the units specified inside a page have to be
shown in the final interface. In particular, operations do not contribute to the layout since they do not show any content. Moreover, each WebML content unit specifies a query whose result includes data retrieved from a single database entity. As a consequence, pages with complex contents are specified as a network of linked units that navigate the data model relationships: some units retrieve data to be shown to the user, while others may be used just for data retrieval support.

The first step of the procedure consists therefore in the identification of the content units that are effectively to show: this subset of the unit of a page is called visualization set.

3Further conditions have been omitted, since they refer to other WebML concepts not introduced in this paper.

- it has an incoming link from another page and only transport outgoing links. In this case, the unit acts as a placeholder for the unit of the coming page to transport its data to its children units (which receive parameters values through the outgoing transport links). The incoming navigational links are instead the only way to pass parameters through different pages (transport links cannot go outside of a page): this constraint of the WebML formalism makes this case different from the previous one;
- it has only outgoing transport links directed to an entry unit: such units are used in WebML to fill the content of form fields and are not shown as separate units.

Figure 4 shows a very complex page model. In the WebML notation transport links are graphically depicted as dashed arrows and navigational links as solid arrows. By applying the above criteria the units highlighted in gray are eliminated from the visualization set. In particular, the data units 1st, 2nd and 3rd are excluded because they have only transport incoming and outgoing links. The other units highlighted in gray represent operations. The remaining units (not highlighted, kept in white) do not satisfy any of the above criteria, so they become elements of the visualization set.

B. Determination of the layout

Once the visualization set has been identified, the units of the page must be positioned in the grid. For this purpose three classes of transformation rules have been identified: pattern-based rules, role-based rules, and relationship-based rules. Each class of rules focuses on a specific aspect of the conceptual models.

Disregard the considered class, a rule is of the form $P \rightarrow L$, where $P$ is an application condition, which is a template page model to test on the currently-analyzed page model, and $L$ is the resulting layout specifying in which way the involved contents have to be positioned in the layout model. It is important to highlight that the application of each rule returns a part of the final layout, i.e., a sub-layout composed only by the units involved in the rule. The last operation of the whole Automatic Display Layout process is the merging of all the produced sub-layouts into the final page layout.

Here after further details for each class of rules are given.

1) Pattern-based rules: In all Web Engineering methodologies the design of hypertexts makes a consistent use of composition and navigational patterns [10], [11]: similar user tasks are implemented with the same content/operation units, defining actually a series of template models, common to all pages performing the same set of operations. This means that the hypertext schemas of that pages are mostly the same, therefore the same models can be reused for pattern identification and contents' placement.

Let us consider a simple example of a modify pattern expressed in WebML. The hypertext fragment in Figure 5 represents the page which allows the editing of an event registration. This operation is available in the administrative area of the application. The page is composed of a data unit Event registration containing the details of a selected

4The focus is on hypertext structure, names readability is not important.
registration and an entry unit Modify registration, representing an entry form for collecting the modified data. The data of the present registration are available in the form fields; they are passed from the data unit to the entry unit as parameters on the link between the two units. When the user submits the data entered in the form (represented by the solid-line link exiting the entry unit) a Modify unit affects the instance of the registration.

Similar patterns can be found in pages that perform the insertion or deletion of data, or the visualization of information by means of subsequent levels of detail. An overview of the WebML patterns can be found in [10].

A first class of transformation rules exploits these patterns by assigning a layout position to the compound units. After the definition of template models, they are applied to the current page to test the adherence to these templates. The positioning of the involved components can eventually be changed at will, but the core of the model template can be used without requiring the developer to starting from scratch for each application.

![Diagram of Modify pattern](image)

**Fig. 5.** The Modify pattern.

Beside the common patterns identified in [10], a Web application may contain also application-dependent recurrent structures. The specification of layout rules can be generalized to any pattern appearing in a Web application.

Figure 6 shows the definition of a pattern-based rule for the Modify pattern of Figure 5. The rule has a left-hand side and a right-hand side. The former represents the hypertext structure a page must contain to apply the rule; the latter represents the corresponding layout in the reference presentation model. Any page containing a data or an index unit, providing content to an entry unit used to modify the fields of the selected object, will have the layout specified on the right-hand side: the data or index unit is displayed at the top, the entry unit at the bottom.

This class of rules produces just a part of the final layout, involving the units that take part in the rule definition. The same is for the other classes of rules. For each page more than a rule is usually applied and each rule produces only a part of the final page layout. Their composition will be done as last step of the whole procedure.

![Diagram of layout transformation rule](image)

**Fig. 6.** A layout transformation rule for the Modify pattern.

2) **Role-based rules:** Once the pattern-based rules have been applied, among the remaining units of the page there could be units having a particular meaning or a higher level of importance with respect to the page content, so that a relevant positioning in the layout of the page is needed. Examples of special contents are: static texts which usually have the function to introduce the reader to the main content of the page, the list of languages in a multi-lingual application, and so on. Such contents usually appear in the same position across the different Web pages. A set of rules can be specified for the layout of these particular contents having a particular role in the global context of the page. Such rules are called role-based rules. Specific roles can be associated with entities (e.g., the entity storing static text or the list of languages) or attribute types (e.g., attributes containing images, or PDF files). These role associations cannot be automatically derived from the data and hypertext models, but must be specified by the designer before the beginning of Automatic Display Layout procedure.

To better explain this point, let us return on the example of the introductory chunks of static text. These text chunks are stored in the database in a dedicated entity, as for all the other data of the application. In the proposed approach the designer can mark an entity as having a particular role, so that the automatic display layout procedure can be aware of the particular role of the data contained in that entity. Units based on marked entities can then be positioned in a particular part of the page (e.g., the first row of the grid of the layout model).

Notice here again that role-based rules are very likely to be reused in different projects, and can be stored, managed and edited for a specific application. Just think on how often static texts appear in a Web application. As another example, consider entities storing contents which need to be treated in a reserved way such as the Latest news, available in many applications.

In the running case, static contents are stored in the entity named Static Contents. To assign a position to the units defined on that entity, the following rule can be defined: Static Contents → (grid, top). It associates the top position in the central grid (in alternative also the left and right column could be specified) to the units defined over entity Static Contents. Figure 7 shows an example of application of such role-based rule: it shows a hypertext instance for which the rule is applicable (the data unit Warning, identified with (1), is based on the entity Static Contents), and its final layout (grid, top).

3) **Relationship rules:** When pattern- and role-based rules cannot be applied to the units of a page, the content can be positioned according to the hypertext structure and the relationships between the entities on which the different units of the page are based.

The content of each WebML page is a set of connected graphs having as nodes the units and as edges the links; each connected graph is treated separately. Among the different units to process in a connected graph, a starting point must be identified: this should be the unit that can be considered the most important within the page. It is called root unit. From the analysis of several real applications developed using WebML, such unit can be identified:
as the destination unit of a (navigational) link coming from the outside of the current page;

• if such a unit does not exist, the root unit is the one having the maximum number of outgoing (navigational and transport) links.

This root unit is usually positioned in the main area of the page. Then, for the remaining units a relative position can be identified, by adopting a heuristic: the hypertext graph is traversed and the source entity of each child unit is compared with the one of its parent unit in the graph. The cardinality of the relationship between the entities over which the units are defined can be used to determine the relative position as follows:

• if the relationship between the entities is one-to-many, the child node is positioned to the right of its parent unit;

• if it is one-to-one, it is positioned in the same area of the parent unit and below the parent unit;

• if it is many-to-one, it is positioned in the left of its parent unit.

Such heuristics have been proven to be valid on several analyzed real industrial WebML applications. Also for this class of rules, they are always reusable in data-intensive Web applications, likely by changing some parameters, such as the positioning of the involved contents which are not forced to be placed in the same manner across different applications.

Figure 7 shows an example requiring relationship rules. The considered graph is the one formed by the units identified with (2) and (3). Indeed, after the application of the criteria in Section IV-A, the remaining units are only the numbered ones: the Student data unit indeed is excluded from the visualization set, while the GetUserOid unit is an operation. Both the not-connected index units Presences and Absences are positioned in the central area (the order of positioning is arbitrary).

C. Automatic layout, manual refinements and ad hoc rules

The layout suggested by the automatic procedure has been evaluated by comparing it w.r.t. the layout manually computed by the designer. At this aim the proposed procedure has been tested on the running case and on several industrial applications developed with WebRatio [3]. The ASP Web Community application is a medium-big Web application: it is globally composed of 700 pages, containing 2866 content units and 952 operation units. Its data model is composed of 48 entities and 190 relationships. Other tested industrial applications are smaller in the number of involved pages, entities, etc., but the results are almost the same. The goodness of layouts are statistically kept and scaled among different applications’ dimensions.

The equivalence of the layout models has been therefore analyzed by means of an automatic procedure. It is quite hard to analyze different layouts. Given a page model, a layout is not univocally determined, so, holding the hypothesis that the same sets of units are positioned, it is difficult to judge the correctness of a layout with respect to another: they could both be suitable for the designed use case. An interesting derivation of this work is the definition of a criterion of aesthetical equivalence, needed to perform layouts’ comparison. It can be stated as follows:

**Definition 1. Aesthetically-equivalent layouts**

Let \( L_1 \) and \( L_2 \) be two different layouts and \( up, down, left, right \) : \( C_1 \times C_1 \) be the relationships stating the relative positioning of two components of the same layout (e.g., \( (a, b) \in up \) means that component \( a \) is in a position one or more rows up component \( b \)). If for each pair of components \( (c_1, c_2) \), with \( c_1, c_2 \in C_1 \cap C_2 \), the same relationships hold for both layouts, \( L_1 \) is aesthetically-equivalent to \( L_2 \).

In other words, two layouts are aesthetically equivalent if the relative positioning of each pair of units is the same, independently on their positioning in the central grid, in the left or in the right columns. The aesthetical equivalence criterion is considered for the acceptance of an automatically computed layout: if its appearance is the same as the one of the manual one in terms of adherence to the aesthetical equivalence criterion, they are considered as indistinguishable. Two aesthetically-equivalent layouts are interchangeable: according to this work, this means that aesthetically-equivalent layouts are treated as if they were identical. Notice that the definition is applicable to contents belonging to both layouts: this is the reason why the set \( C_1 \cap C_2 \) is considered as the aesthetical equivalence test: the intersection retrieves only the contents common to both layouts.

Table I shows the results obtained by applying the procedure on the running case. Column All rules shows the results obtained by applying the three classes of rules. Column No patterns shows the results obtained without applying pattern-based rules; finally, column No roles does not consider the role-based rules. Notice that in the most part of the cases about the 70%-80% of the pages of each area are rendered in a similar manner w.r.t. manual layouts and according to the aesthetical criterium previously defined.

Although for the major part of the pages we would like to obtain the most likely layout, there will be a set of pages whose layout is not the most appropriate one (e.g., this is the case of complex hypertext pages). Therefore a subsequent hand-done refinement is allowed, but the gain is that the developer can start from the “draft” layout, rather than from scratch.

![WebML model](image1.png) ![Computed presentation model](image2.png)

Fig. 7. Application of the role-based rule.
TABLE I
RESULTS OF THE RUNNING CASE SITE TEST. THE TOTAL NUMBER OF PAGES OF THE SITEVIEW IS INDICATED IN BRACKETS.

<table>
<thead>
<tr>
<th>Siteview (Pages)</th>
<th>All rules</th>
<th>No patterns</th>
<th>No roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students area (134)</td>
<td>81.12%</td>
<td>73.34%</td>
<td>38.34%</td>
</tr>
<tr>
<td>Teacher area (464)</td>
<td>80.25%</td>
<td>72.56%</td>
<td>75.44%</td>
</tr>
<tr>
<td>Administrative area (102)</td>
<td>90.10%</td>
<td>88.23%</td>
<td>87.25%</td>
</tr>
</tbody>
</table>

V. IMPLEMENTATION OF THE TRANSFORMATIONS AND DISCUSSION

The transformations presented in the previous sections have been implemented in the optics of an easy integration in the WebRatio CASE tool [3]. This tool saves all the conceptual models by internally representing them in an XML format. Since the beginning of the work, the procedure has been thought for final users, which are not necessarily experienced developers. Therefore, there was the need to support this kind of users in the specification of the rules for the Automatic Display Layout procedure without knowing much about Model Transformation languages. Initially, we have implemented the whole procedure using a classical programming language (Java) and we have tested the feasibility of our approach on several industrial applications. We are now evaluating the use of graph grammars (GG) for expressing the same transformations, due to the immediate mapping of the WebML conceptual models into graphs. The current implementation based on graph grammars considers only the basic pattern and relationships layout rules. To be able to use graph transformation for XML documents, they must be translated into graphs. For this task we based our work on the simple solution provided in [12].

VI. CONCLUSIONS

In our proposal graph transformations have been introduced to specify transformations between conceptual PIMs with the specific aim of performing the automatic layout of the content of Web application pages. As discussed in the previous section, the specification of rules based on GG provides a good support in the modification of the predefined rules, which can be defined also by Web designers. A visual support in the specification of the actual implementation of the transformation rules is desirable, to simplify possible changes in the transformation model’s rules by the final user.

REFERENCES