Improving e-Form Layout Through Analysis of Form Semantics and Validation Checks

C. Vassilakis[1], G. Lepouras[1], S. Rouvas[1], P. Georgiadis[1]

[1]e-Gov Lab, Dept. of Informatics and Telecommunications, University of Athens, Greece
{costas, gl, rouvas, p.georgiadis}@e-gov.gr

Abstract. Transaction services offered by public authorities vary from simple forms with few fields to multi-form compound documents with hundreds of input areas. In the latter case, field placement within forms is of particular importance for facilitating the filling and error correction processes. In this paper we present an approach to improving the form layout by exploiting validation checks that are usually associated with electronic forms, as well as semantic information that may be attached to form fields by designers.

1. Introduction

Most of the interactions of citizens with public authorities are performed through forms, which may range from simple documents with less than ten fields, such as a statement for address change, to highly complex document sets, such as tax return forms or social benefit claims. Form layout and field placement is significant for providing easy-to-use forms. Even in simple forms, using consistent layout across forms allows users to familiarise more quickly with the forms and exploit the knowledge amassed from using one service in the context of other services [1]. In complex multi-form services, layout and field placement are of higher importance, since (a) apposite document structure aids users in locating the fields they need to use and (b) placing conceptually related fields closely together facilitates the process of gathering the appropriate data from relevant documents and crosschecking the values.

In this paper we present an approach that integrates all phases of electronic service development, including definition of form fields and their semantics, form layout and validation checks. This information is then exploited during an optimisation phase for improving the electronic service layout. The platform also encompasses generic user interface design best practices and policies (e.g. maximum number of elements in a single form), providing thus a holistic solution to user interface design for e-services.

The development environment

To support the process of form layout optimisation, a prototype of an environment has been built that supports all aspects related to the development of transactions services, which are described in the following paragraphs:

1. *Definition of the form elements comprising the electronic service.* This facility allows the designers to define all form elements that may be needed in the context of the electronic service, such as form fields, associated text labels, form headers/footers and navigation controls. A number of the details entered for each field are not directly related to the issue of optimising the layout of the forms involved in the transaction service, but are required for other aspects of the electronic service development, which are supported by the development environment.
2. *Designation of semantic axes*. Semantic axes are *thematic categories* under which form fields can be classified. For example, for modelling a tax return form, candidate semantic axes include, amongst others, *income, expenditure, pre-paid taxes, salaries, real estate* and *informational*. Each field within the service may be assigned to any number of semantic axes – for instance, the field in which income from salaries is filled in can be assigned to the semantic axes *income* and *salaries* while the field representing the pre-paid taxes from stipendiary occupation falls under the axes of *salaries* and *pre-paid taxes*. Semantic axes are thus *candidate form areas* (for paper forms). Determination of the most suitable semantic axis to use for each field in the final layout is discussed in section 3.
3. *Definition of validation checks*. Validation checks are part of the business rules governing the electronic service, specifying conditions that values entered by the users must fulfil. Validation checks are important for determining optimal field placement, since –as stated in section 1– it would be beneficial if fields involved in the same validation check appeared close together within the final form layout. Within the development environment, service developers (domain experts and IT staff) enter validation checks through an editor, supporting the following types of checks: (a) *A Requires B* (if a value is entered in field *A* then a value must be entered in field *B*) (b) *A Precludes B* (if a value is entered in field *A* then field *B* should be left blank) (c) *A cmp B * c*, where *A* and *B* are form fields, *cmp* is a relational operator ($=, \neq, >, \geq, <, \leq$) and *c* is a constant value. This validation check category allows for modelling of arithmetic constraints on form fields and (d) *Custom check.* This category of validation checks is used to model complex constraints that does not fall in groups (1) – (3). For these checks, domain experts may only specify the fields involved in the check, while IT staff supplies the code.

   This categorisation scheme allows domain experts to enter all validation checks falling into the first three groups through a graphical, environment; the code is generated automatically. Finally, a *weight* is associated to each validation check, specifying how important is to keep the fields involved within this check closely together. The value of the weight (in the range 1-100) is determined by the domain experts based on their experience, regarding the number of documents usually failing this validation check, the number of citizens using any of the form fields involved in the check etc.
4. *Specification of layout constraints*, i.e. designation of options such as (a) the maximum number of fields that may be placed in a single web page, (b) the maximum number of pages that should be used for placing the various fields and (c) whether two or more distinct semantic axes may be placed in a single web page.

It should be noted that some of these goals are often contradictory; for instance, in order to minimise the overall number of pages within a service, the number of fields

per page must be increased. Service designers should determine a “golden mean” between contradictory goals, to produce a suitable layout.

Computing an optimal form layout

Once the appropriate information has been entered into the system, the process of computing an optimal form layout may begin. The objective of this procedure is to split the fields required for the electronic service into a number of web pages that (a) contain conceptually related fields (fields are considered conceptually related if they have been assigned to the same semantic axis) (b) cluster fields interrelated with validation checks on the same page whenever possible (c) keep the overall number of fields within a single page below the limits specified in the form layout constraints and (d) minimize the total number of pages.

In order to compute a solution with the above characteristics, the system constructs an undirected graph G = (V, E), whose vertices *v* are the fields that appear within the electronic service. Two vertices v_1 and v_2 representing fields f_1 and f_2 are connected with an edge *e* if there exists a validation check *VC* that relates the values of f_1 and f_2. For each edge, a weight *W* is assigned, which is equal to the weight assigned by the domain expert to the validation check. If there exist multiple validation checks VC_1, VC_2, ..., VC_n involving fields f_1 and f_2 and having weights W_1, W_2, ..., W_n, respectively, then fields f_1 and f_2 are connected with a single edge *e* whose weight *w* is equal to the sum of the individual weights. The objective of the optimisation algorithm is to partition the vertex set *V* into mutually disjoint subsets V_1, V_2, ..., V_m, such that the cost of the weights of all edges interconnecting vertices belonging in different subsets is minimised; the costs of edges connecting vertices within the same vertex subset is disregarded. An additional constraint for vertex subsets is that for each such subset, all vertices (fields) included in this subset should be assigned to the same semantic axis S_i; however, fields assigned to the same semantic axis are not placed necessarily on the same vertex subset, i.e. a semantic axis may be split in multiple vertex subsets.

The vertex subsets V_i will actually be the different web pages comprising the electronic service. Intuitively, the cost of the edges connecting vertices (fields) in different subsets (pages) is a measure of the extraneous navigation actions that users will perform for the purpose of looking up values of fields that have been placed on different pages. The constraint of formulating vertex subsets V_i with fields belonging to the same semantic axis guarantees the semantic affinity of web pages.

According to the description presented above, the task of optimising the layout of a transaction service is isomorphic to the graph partitioning problem ([2], [3]). Software libraries for solving graph partitioning problem have become available; in the prototype environment we used the hMETIS package ([4]) which directly supports n-way graph partitioning, formulates high-quality partitions and is very efficient, even in low-end workstations. Once the optimal form layout has been computed, it is presented to the user for inspection. The user is able to perform modification to the proposed layout or directly request the generation of the respective HTML pages. The generated HTML pages may be finally processed by HTML experts and/or by specialised software to provide for the final aesthetic touches.

A case study: the Greek Tax Return Form

In order to validate the proposed approach, an experiment was set up, using the Greek tax return form as a case study, which includes approximately 800 fields broken down into 12 thematic areas. The electronic version encompasses 195 validation checks, in which the values of 503 declaration fields are correlated.

In the current electronic service layout, 49 validation checks (25.1% of the overall number) involve fields that have been placed on different web pages. From statistic analyses, it has been determined that these 49 validation checks account for the 54% of the errors detected in electronic submissions, requiring thus the users to issue two (or more) requests for web pages, in order to correct the errors.

The layout proposed by the system included a change of the semantic axes, by entirely abolishing two of them, and distributing their fields in four others. One semantic axis was replaced by three, more specialised ones (this is actually equivalent to retaining the original axis and using three pages for placing its fields) and 18 fields were moved to a different web page, since they were assigned to the pertinent semantic axis *and* were more tightly coupled (through validation checks) with the fields of the page they were moved to. In total, the number of web pages within the service remained constant, the number of validation checks involving fields from different web pages dropped to from 49 to 34, accounting statistically for the 24.3% of the total errors (from the initial value of 54%).

Conclusions

In this paper we have presented a scheme for improving the form layout of complex electronic services, by exploiting semantic information that is attached to form fields by designers and validation checks. A prototype for a development environment has been created into which domain experts and IT staff enter information regarding the needed fields, semantic axes, validation checks and layout constraints, and the system automatically generates an optimal layout for the service.

References

1. New York State Office for Technology, “Forms Design Issues for Electronic Document Management Systems”, available at http://www.oft.state.ny.us/cookbook/8_forms.htm
2. C. E. Ferreira, A. Martin, C. C. de Souza, R. Weismantel, and L. A. Wolsey. “The node capacitated graph partitioning problem: a computational study”, Mathematical Programming, 81, pp. 229-256, 1998.
3. P. Kuntz, “A distributed heuristic for finding clusters in vertex sets”, International Symposium on Mathematical Programming, Lausanne, 1997.
4. hMetis project, “Hypergraph Circuit Partitioning”, http://www-users.cs.umn.edu/~karypis/metis/hmetis/