Re-engineering legacy Web applications into RIA s by aligning modernization requirements, patterns and RIA features

José M. Conejero *, Roberto Rodríguez-Echeverría, Fernando Sánchez-Figueroa, Marino Linaje, Juan C. Preciado, Pedro J. Clemente

Quercus Software Engineering Group, Department of Computer Science, University of Extremadura, Av. Universidad s/n, 10071 Cáceres, Spain

ARTICLE INFO

Article history:
Received 9 April 2012
Received in revised form 8 March 2013
Accepted 15 April 2013
Available online 30 May 2013

Keywords:
RIAs
Modernization
Model driven web engineering

ABSTRACT

Rich Internet Applications (RIAs) have become a common platform for Web developments. Its adoption has been accelerated thanks to different factors, among others, the appearance of patterns for typical RIA behaviors and the extension of different Model Driven Web Engineering methodologies to introduce RIA concepts. The real fact is that more and more developers are switching to RIA technologies and, thus, the modernization of legacy Web applications into RIA s has become a trend topic. However, this modernization process lacks of a systematic approach. Currently, it is done in an ad hoc manner, being expensive and error-prone. This work presents a systematic process to modernize legacy Web applications into RIA s. The process is based on the use of traceability matrices that relate modernization requirements, RIA features and patterns. Performing some operations on these matrices, they provide the analyst with the necessary information about the suitability of a pattern or set of patterns to address a given requirement. This work also introduces two measures, the degree of requirement realization and the degree of pattern realization, which are used to discuss the pattern selection. Finally, the applicability of the approach is evaluated by using it in several Web systems.

© 2013 Elsevier Inc. All rights reserved.

1. Introduction

Rich Internet Applications (RIAs) have emerged as the most promising platform for Web 2.0 development combining the lightweight distribution architecture of the Web with the interface interactivity and computation power of desktop applications (Fraternali et al., 2010b). This is the reason why more and more developers are switching to RIA technologies. This fact has brought about the appearance of two trends.

On the one hand, Model Driven Web Engineering (MDWE) approaches have been extended with new modeling primitives to obtain the benefits provided by RIA features. These approaches have followed two different paths: improving the capability of dealing with data and business logic (Bozzone et al., 2006; Ginzburg et al., 2009; Toffetti et al., 2007) and strengthening presentation management (Linaje et al., 2007; Meliá et al., 2008; Uribeta et al., 2007). On the other hand, there is a growing trend in the industry to perform a modernization of their legacy Web Applications (WA) introducing RIA features (Mesbah and van Deursen, 2007; Rossi et al., 2008; Samir et al., 2007). This trend is even more evident with the transition to the forthcoming HTML5 standard, gaining momentum.

Although Web development approaches and techniques have properly evolved to support RIA capabilities, the real fact is that there is a huge amount of legacy non-model-based Web applications in the industry that were developed without considering RIA features and their migration to the RIA domain is being performed in an ad hoc manner, leading to very expensive and error-prone processes. Moreover, the adaptations performed related to RIA features are not reusable since they have to be applied again from the scratch in any new migration process. This is a significant drawback since there are quite a few adaptations that recurrently appear in many RIAs (Koch et al., 2009), e.g. synchronization patterns needed to work in a disconnected mode. This is the reason why the software industry has been demanding more and more some kind of systematic helps to face this modernization process.

This claim motivated the MIGRARIA project (Migraria-Project, 2012; Rodríguez-Echeverría et al., 2011), where a systematic and semi-automatic process to modernize legacy non-model-based data-driven WAs into RIAs has been defined (Fig. 1). This project consists of two main steps: reverse engineering to obtain a model-based specification from the legacy WA (step a in Fig. 1); and forward engineering to combine RIA patterns with the obtained model to fulfill the new appearing modernization requirements. The forward engineering process is based on the utilization of traceability matrices that relates modernization requirements, RIA features and patterns. The process can be divided in 3 steps (b–d in
The output of step b is a matrix representing the RIA features affected by the appearing modernization requirements. For each new requirement, the set of RIA features to be taken into account are marked (i.e., processing at client side, partial refreshing, etc.). Step c suggests a set of RIA patterns to be used for implementing the new modernization requirements. Once the RIA patterns are selected, a weaving process is used to introduce their functionalities into the legacy Web application (step d).

In previous works we have shown a general overview of the approach (Rodríguez-Echeverría et al., 2011) and detailed the reverse engineering process. While in Rodríguez-Echeverría et al. (2012) it is described how to obtain models from a non-model based legacy application (step a in Fig. 1) based on the Struts framework, the work in Rodríguez-Echeverría et al. (2010) described the application of RIA patterns to a legacy system by means of model weaving (step d in Fig. 1).

This paper focuses on steps b and c of the forward engineering process (dashed rectangle in Fig. 1), being its main contribution covering the existing gap between the definition of the modernization requirements and the architectural design. It complements our previous works in Rodríguez-Echeverría et al. (2011) and Rodríguez-Echeverría et al. (2010) by: (i) providing a concrete instantiation of the forward engineering step; (ii) defining a way to identify the more suitable patterns to be applied to the legacy system from the RIA modernization requirements, respectively. For this purpose, the process described in this paper takes as input the models generated by the steps described in Rodríguez-Echeverría et al. (2012) (step a in Fig. 1) and provides the input for the steps described in Rodríguez-Echeverría et al. (2010) (step d in Fig. 1).

The rest of the paper is as follows: Section 2 summarizes RIA features and shows how some RIA modernization requirements appear in a motivating example. Both are the entries of process b in Fig. 1. Section 3 is devoted to show how RIA features relate to requirements (step b in Fig. 1) and patterns (step c in Fig. 1) and how to measure the suitability of patterns for fulfilling requirements. Section 4 shows the process followed to evaluate the approach. Finally, Section 5 discusses related work while Section 6 presents conclusions, summarizing the main benefits and limitations of the approach.

2. RIA features and modernization requirements

This section presents a conceptual framework to assist the Web analyst to elicit modernization requirements conveying RIA features. This framework mainly proposes a concrete organization of well-known RIA features (Bozzon et al., 2006; Brambilla et al., 2010; Fraternali et al., 2010b; Melià et al., 2008, 2010; Preciado et al., 2005) from a conceptual viewpoint. It is worth noting that the definition of the framework at a high level of abstraction enables RIA features to be reused in different modernization projects.

First we briefly introduce the conceptual framework conformed by RIA features. Then, and through a motivating example, some modernization requirements are elicited considering some issues of the legacy application. Both artifacts, requirements and RIA features, are the entry points of process b in Fig. 1.

2.1. RIA features

The conceptual framework proposed arranges the RIA features attending to the RIA conceptual layers: data, business logic, presentation and communication. RIA features are specified in a declarative manner to maintain them independent of any RIA technology.

Data. RIA technologies support client-side storage that can be either volatile or persistent. This scenario leads to four different possibilities for data allocation and duration: non-persistent data at the client side, persistent data at the client side, non-persistent data at the server side and persistent at the server side.

Business logic. In RIAs part of the application business logic can be moved to the client, leading to different possibilities for business
logic computation: client side, server side and mixed. Additionally, RIA often exploit a single-page paradigm for the interface organization. This fact brings different possibilities for page/components loading: initially; incrementally (at each interaction event sub-pages or components may be added or removed); with refresh (at each interaction event sub-pages or components may be refreshed).

**Presentation.** RIA introduce the single-page paradigm that allows displaying the information without switching the UI context among different pages/subpages. The interface can also be extended dynamically, by loading parts of the presentation logic and interface elements at run time (i.e. partial refreshment). The number of controls and containers is wider than in purely HTML-based interfaces and the dynamic customization of widgets allows them to be adapted to specific contexts. Extended interaction and temporal behaviors further improve the user experience.

**Communication.** In RIA, communication can be originated at client or server side and can be asynchronous. As data and business logic are distributed, communication may involve data transfer flows between the client and the server, as well as control flows to synchronize the business logic between the two tiers of the architecture.

These features are summarized in Table 1 (labeled F1 to F19). They can be found in many Web applications combined in several ways, leading to different RIA scenarios. For example, with a proper combination of these features one could obtain disconnected work. However, this scenario can also be obtained with different combinations of these features. Considering again disconnected work, persistent client-side data and partial refreshment (F2 and F14) should be mandatory. However, data granularity for communication (F15 and F16) should be chosen based on the amount of information and frequency of transfer, and this varies from one application to another.

Identifying scenarios like this is not the aim of this paper. One can obtain many of them from the literature (Koch et al., 2009; Mesbah and van Deursen, 2007; Rodríguez-Echeverría et al., 2010; Toffetti et al., 2007; Urbieto et al., 2007; Wright and Dietrich, 2008). Our first aim at this step is providing the Web analyst with a conceptual framework to assist on the decomposition of modernization requirements into RIA features. The conceptual framework may play an important role as a domain specific reference and a common language among modernization engineers. Obviously, Web analyst experience and expertise is key to make such decomposition in a successful way.

### 2.2. Modernization requirements: motivating example

In order to illustrate the work presented herein a case study about the modernization of a ticketing Web application is partially devised. In concrete, a ticketing e-commerce application specialized on events, such as concert and theater plays, sport events and so on, is considered. The rationale behind this modernization is basically to increase sales from the owner company point of view. That company has decided to make an inversion to solve the main issues of the current system. Roughly, three main goals are pursued: (1) improving the efficiency and user experience of the purchasing process; (2) providing the user with event recommendations and personal discounts and offers; and (3) improving the overall usability of the system.

Concerning the first goal, the current purchasing process is based on a seat pre-booking mechanism, implemented by a plain Web form. The normal purchasing process consists of 3 steps: (1) the user selects the event; (2) the user pre-books an specific amount of available numbered seats on a concrete venue area (pre-booking page); and (3) the user confirms her booking by accomplishing the purchase process (summary page). Once the user has selected the event, a plain Web form is presented to select the seat location inside the event venue. Then, the user may pre-book those seats by clicking a button. A summary Web page is then presented with a counter indicating the time limit to accomplish the purchase. The user may confirm her booking by clicking the purchase button.

Note some issues arising in this solution. (1.1) When a seat selection conflict (a seat pre-booked twice) appears, the affected user is notified in the summary page and she must return to the pre-booking page to change her selection, degrading the user experience. (1.2) The pre-booking page shows the available seats on load time, so this information is only updated on page reload. Then the user has not got live information about seat availability. (1.3) The whole process is performed at server side while part of the business logic could be run at client-side reducing unnecessary client-server roundtrips and server workloads. That last issue gets special relevancy when tickets for events with heavy demand are put up for sale. A bad performance in that moment may produce unsatisfied users and, therefore, a decrease in sales. So responsiveness is clearly a significant non-functional requirement for the system.

The addition of RIA features to the purchasing mechanism may solve the aforementioned issues providing the user with a better user experience and hence increasing sales. In this specific case, partial refreshment, processing and storing capabilities at client side and asynchronous communications come to the scene. In particular, it is suggested to change the plain Web form (pre-booking page) for an interactive map of the event venue (see Fig.2). That map will represent the layout of the event venue and will show live information about seat status (see Fig. 3). Additionally, it will provide the user with information on-demand regarding the amount of free seats, prices and package offers on each zone of seats.

Concerning the second and third goals (event recommendation and discounts and usability), it is worth noting two more issues recurrently appearing in legacy e-commerce applications...
developed with traditional Web technologies. As some studies (Nielsen et al., 2001) state, users usually get frustrated on e-commerce applications because they do not find what they are looking for. Although there is a wide range of possibilities, some recurrent causes may be: misspelled text on text input fields for searching (complex interaction); user interfaces with high information density; and complex product catalogs to access to the most demanded ones. (1.5) And finally, a high interactive user interface (fluid and responsive) (Escalona and Koch, 2006) helps to reduce the amount of shopping carts that are discarded.

In this case, some RIA features may alleviate the aforementioned issues by making proper suggestions to the user or providing her with personalized information. In particular, it is recommended to allow the user navigate interactively through a gallery of top recommended events (most visited, high rated, upcoming, etc.). Regarding usability, auto-completion searching text fields or interactive calendars for date-based event search will be used in order to avoid misspelled searching strings. Finally, the different functionalities of the system will be accessible through a menu where each menu item includes different sub-options that are presented to the user on demand (e.g. when the user moves the mouse over a menu item).

Considering those issues, the modernization requirements (input for process b in Fig. 1) to be added to the original system are the next (note that each requirement is related with the issue that originated it):

- **RQ1**: The purchasing mechanism must be as efficient as possible, specially when events with heavy demand. It will include a timer to reduce the time a seat could be pre-booked in order to improve efficiency and user experience (1.1 and 1.3).
- **RQ2**: A map must present live information about the seat status avoiding seat pre-booking conflicts. The map information will be updated without refreshing the whole page (1.1 and 1.2).
- **RQ3**: The user must be provided with additional information (prices, especial offers, etc.) about a free seat for any zone. This functionality must be achieved reducing the roundtrips with the server as much as possible (1.3).
- **RQ4**: The system must have a top recommended section where the user may navigate through the most visited events by means of graphical information (1.4 and 1.5).
- **RQ5**: The system must assist the user to input data on searching text fields by proposing possible values according to the text already written (1.4).
RQ6: The system must assist the user to search for events by date, avoiding page refreshment, reducing the round-trips with the server and loading only the data related to the month that the user is watching (I.4).

RQ7: The system must provide the user with an interactive menu to access easily to the main functionalities. The navigation through the menu must be efficient in terms of responsiveness (I.5).

As a real example of our case study, Figs. 2 and 3 present snapshots of a ticketing system\(^2\) to partially illustrate the modernization requirements introduced before.

### 3. Patterns selection

This section depicts the process followed to select some RIA patterns that cover the RIA functionalities to be added to the system. The process is based on building two traceability matrices that relate requirements with RIA features and patterns with RIA features, respectively. While the former is built for any modernization process, the latter is a fixed matrix shared by all the modernization processes and contains patterns extracted from Brambilla et al. (2010), Rodríguez-Echeverría et al. (2010), Vora (2009), Yahoo!-Patterns (2011). Then, based on some matrix operations and the definitions of some metrics, the patterns to be used are suggested.

#### 3.1. Mapping requirements to RIA features

The analyst has the responsibility of deciding which RIA features must be applied for each new requirement. Based on the right understanding of these features and his own expertise, the analyst must decide which the most suitable set of RIA features for a given requirement is (process b in Fig. 1). This section shows how this decision may be supported by a special kind of traceability matrix that relates modernization requirements with RIA features.

The relations between requirements and RIA features are stored in the requirements traceability matrix (rtm – Table 2). Rows in the rtm represent the requirements while the columns are filled with the RIA features identified in Section 2.1. For a given requirement the marks indicate the RIA features that must be considered for its later design. Next, we illustrate how rtm is obtained in our running system. Only RQ1 will be discussed for the sake of simplicity. However, a similar discussion can be followed for the rest of requirements obtaining the traceability matrix of Table 2.

RQ1 requires modifying the purchasing method, whose main implementation element is a shopping cart. Obviously, each user owns her shopping cart so this entity demands client-side storage. Moreover, when the user confirms the cart, the system creates a persistent order at the server tier for data consolidation. This situation requires data replication at client and server tiers (features F1 and F4), due to seats order must be permanently stored at server tier. Regarding business distribution, the selection of seats, the calculation of the total amount to be paid and the shopping cart management is performed at client side. However, the final purchasing process and the order creation are performed at server side. Thus, mixed business logic computation is required (F7). Note that local processing is the main approach but server processing is required for seats status synchronization and order purchase. Moreover, all the data involved should be loaded initially, so that F8 is required. Finally, note that all the cart updates at client side must be propagated to the other side, in other words, the cart items must be synchronized between client and server so that features F15 and F17 are required. In particular, when a seat is pre-booked, a tuple-based communication will be required (since single data is modified). The confirmation of the whole shopping cart will also require communication between client and server, however, in this case the total seats will be atomically sent to the server so that dataset communication is required (feature F16).

#### 3.2. Mapping patterns to RIA features

There are different RIA pattern collections to deal with recurrent Web problems such as the example situations described in Section 2.1 or the requirements described in Section 2.2. Examples of these patterns are those introduced in Brambilla et al. (2010), Vora (2009), Yahoo!-Patterns (2011), to deal with presentation issues, or those introduced in Rodríguez-Echeverría et al. (2010) that are focused on synchronization matters. Similarly to the case of requirements we can obtain a matrix showing the RIA features
involved in these patterns. This new traceability matrix is called the **patterns traceability matrix (ptm)**. This matrix is taken as input by the step *Matrices Matching* in our approach (process c in Fig. 1). As aforementioned, this matrix is reused in all the modernization processes since it is previously filled in with patterns extracted from Brambilla et al. (2010), Rodriguez-Echeverria et al. (2010), Vora (2009), Yahoo!-Patterns (2011). Nevertheless, instead of showing the ptm with the whole set of these patterns, we use a significant subset of them to illustrate the different cases when doing the matching between ptm and rtm in the running example.

Before showing the ptm, we will describe the patterns that have been used to illustrate our process: Carousel, that allows a graphical navigation between a set of items in a limited space (Vora, 2009; Yahoo!-Patterns, 2011); the **Dynamic tooltips on page data** (Brambilla et al., 2010) to show context-based information on a particular region; **Partial page refresh** (Brambilla et al., 2010), that allows some portions of the page to be reloaded without loading the whole page; **Module Tabs** (Yahoo!-Patterns, 2011) or **Left Navigation Bar** (Yahoo!-Patterns, 2011), both used to present the user an interactive navigation through a set of options (menu) where the items are sensible to user’s events.

Regarding distribution concerns, the patterns introduced in Rodriguez-Echeverria et al. (2010) identify several synchronization scenarios to deal with the management of shared resources, collaborative work, and typical client to server data synchronization. Examples of these patterns are: the **Client Reconnection**, used by systems that allows disconnected work when a client reconnects; the **Data Event Notification** pattern, that provides a mechanism for asynchronously notifying the connected clients about an event produced in the server; the **User Triggered Action Replication**, used when an action performed at client tier must be replicated at the server; the **User Triggered Bulk Data Replication**, used to synchronize a set of data between clients and server.

Table 3 presents **ptm**. Here the rows represent the RIA patterns. A cell with “o” indicates that the pattern could be also applied by optionally using the RIA feature of this column. As examples, next we explain how the rows for the **Data Event Notification**, **User Triggered Bulk Data Replication** and the **Partial Page Refresh** patterns are obtained. The **Data Event Notification** pattern (P2) (Rodriguez-Echeverria et al., 2010) allows the asynchronous communication between server and clients. To make the paper self-contained, Fig. 4 presents a snapshot of the structure of the pattern. Since the pattern was defined in WebML (Ceri et al., 2002), this figure is presented using this notation. The pattern is based on the RIA extensions to WebML introduced in Bozzon et al. (2006) and Toffetti et al. (2007). The entities *NotifyEvent* and *ReceiveEvent* introduced in Toffetti et al. (2007) are used for the broadcasting and the reception of an event, respectively. Observe in Fig. 4a that this pattern requires an entity (*NewEvent*) to store the event at server side (marked “S” in the upper left corner of the entity). This is why feature F4 is required. The pattern behavior is also divided into two different parts. The first part performs the broadcasting of the event (Fig. 4b). Observe that all the entities involved in this part of the pattern are processed at server side, as indicated again by “S”. The second part of the pattern (Fig. 4c) is dedicated to the reception of the event. In this case, the entities involved in this part are processed at client side (note the “C” represented in the entities). Due to this business logic distribution between client and server, the mixed **RIA distribution** feature (F7) is required by this pattern. Finally, regarding the communication between tiers, the pattern allows the communication from server to clients (F18). Moreover, as suggested in Toffetti et al. (2007), the data entity that stores the event information (*NewEvent*) may be used to send information to the clients. Although this information is often a single field (*tuple*), optionally a data collection (*dataset*) could be also sent (F15 and F16).

Regarding the **User Triggered Bulk Data Replication** pattern (P4) (Rodriguez-Echeverria et al., 2010), Fig. 5 shows the model for this pattern (allows sending a set of data to the server so that these data may be synchronized at client and server sides). The pattern is defined in terms of the **XMLOut** and **XMLIn** entities to marshall and unmarshall a set of data (again RIA extensions of WebML). As it is suggested by the guidelines defined in Bozzon et al. (2006), this synchronization usually requires the duplication of the data entity which stores the data to be synchronized at client and server sides (sometimes with different names, e.g. Order and CartItem entities in our running example). Note that the pattern requires functionality to be executed at server and client sides (F7) and data to be stored at both sides (F2 and F4 and, optionally, F1). Moreover, the data involved should be initially loaded at client side so that F8 is required. Finally, client
must send the set of data (F16) to be replicated at server side (F17).

The Partial Page Refresh pattern (P5) (Brambilla et al., 2010), defined in WebML, uses a new type of link introduced in WebRatio \(^3\) (AJAX link) that allows the refreshment of entities linked without affecting other independent entities of the page (see Brambilla et al., 2010 to obtain more details). Obviously, the pattern is related with the partial refreshment feature (F14). However, it also requires that the client processes the refreshment of the entities pointed out by the new link. This is why client side processing is also required (feature F5). Since the data to be refreshed is provided by the server on demand and then stored at client side, Volatile client data and With refresh computation time features are required (F1 and F10, respectively). Finally, the pattern requires the features Look&Feel and Interaction behavior (F11 and F12) to manage the presentation and behavior options of the component being refreshed.

The same reasoning can be followed for the rest of patterns. It is important to remark again that this process is done only once and, thus, ptm is reused among different modernization processes. Here we have shown how it is partially obtained for illustration purposes. Once requirements and patterns are aligned with RIA features it is turn for matching requirements and patterns as shown in next section.

3.3. Requirements and patterns matching

This section presents a process whose main goal is to bridge the gap between the new modernization requirements and RIA architectural design since parameterized solutions (patterns) for the different requirements are suggested. The process is based on a matching between the two traceability matrices presented in previous sections (rtm and ptm). This matching is based on simple mathematical operations that take as input the values obtained in the matrices (Matrices Matching in Fig. 1). The process is computed by a simple tool (based on spreadsheets) that performs the matrices matching and provides the suggested patterns according to the analysis of the results obtained.

In order to make our traceability matrices computable, they are transformed into binary matrices by converting any cell with “x” or “o” into a 1 and blank cells into 0. The binary matrices obtained for the rtm and the ptm in our running system are depicted in Tables 4 and 5, respectively.

Our process performs the matrix product of the first binary matrix by the transpose of the second one, obtaining the correspondence matrix (cm) that shows the number of RIA features shared by requirements and patterns (Table 6).

Using these values, we may infer that the patterns P6 and P7 are candidate for the design of the requirement RQ4 since they are the patterns that share more RIA features with the requirement (even P7 is better positioned than P6). However, these values could also lead to invalid conclusions since they may be influenced by the granularity of the requirements, e.g. requirements defined with a low cohesion level where several concerns or functionalities are involved may lead to suggest too many patterns for the same requirement. In that sense, we encourage to use fine granularity level requirements. Anyway, to minimize the impact of this issue, the values obtained in the matrix are normalized by relating them with the number of RIA features required by the corresponding requirement and pattern. We called these relations Degree of

\(^3\) WebML case tool.
requirement realization (DRR) and Degree of pattern realization (DPR). The DRR for a requirement \( RQ_i \) with respect to the pattern \( P_j \) is defined as:

\[
\text{Degree of requirement realization (RQ) } (P_j, P_i) = \frac{cm_{ij}}{\sum_{k=1}^{n} rtm_{ik}}
\]

(1)

On the other hand, the DPR for the requirement \( RQ_i \) with respect to the pattern \( P_j \) is defined as follows:

\[
\text{Degree of pattern realization (RIA) } (P_j, P_i) = \frac{cm_{ij}}{\sum_{k=1}^{n} ptm_{ik}}
\]

(2)

where \( cm_{ij} \) represents the cell \([i, j]\) of the correspondence matrix, \( |F| \) represents the number of RIA features (19 in our case) and \( rtm_{ik} \) and \( ptm_{ik} \) represent the cells \([i, k]\) and \([j, k]\) of the requirements traceability matrix and the patterns traceability matrix, respectively.

A value of 1 in DRR indicates that all the RIA features demanded by the requirement are supported by the pattern (although it could also support other ones). A value close to 0 indicates that the pattern just supports a few of the RIA features needed by the requirement while a value of 0 indicates that the requirement and the pattern do not share any feature. Similarly, a value of 1 in DPR denotes that all the RIA features supported by the pattern are needed by the requirement (although it could also require other ones). A value close to 0 in DPR denotes that an insignificant amount of the RIA features supported by the pattern are required by the requirement. Table 7 shows the different combinations that may exist in terms of the values of these metrics for a particular pattern and requirement. \( RQ \) represents the set of features required by a requirement while \( P \) denotes the set of features covered by a pattern. We are interested in those patterns with highest values in DPR and DRR. The best situation is illustrated by cell \((P \geq RQ)\) in Table 7.

In order to automatize the patterns selection, an algorithm is used to compute the metrics and help the developer at selecting the most suitable patterns. Based on this algorithm, three different situations appear: (case i) a requirement fulfilled by a pattern, (case ii) a requirement that may be fulfilled by the combination of two or more patterns, (case iii) a requirement where two or more different patterns may be applied in an exclusive way so a selection must be done.

The algorithm is illustrated in Fig. 6. For a better understanding, the functionality is summarized as follows:

- **Step1.** The algorithm selects for each requirement the patterns with highest value of DPR.
- **Step2.** Then the value of DRR for the patterns selected is evaluated. Those with a DRR of 1 are selected as candidates. At this step, the algorithm allows the distinction between the three different cases described above:
  - **Step2.1.** If there is just one pattern with value of 1 in DRR, then it is the only candidate (case i).
  - **Step2.2.** If there are two or more patterns with a value of 1 in DRR, then the developer must select one of them as candidate (case iii).
  - **Step2.3.** If there is no pattern with value of 1 in DRR, then the patterns firstly selected could be combined (case ii) or just one of them should be selected as candidate (case iii). This situation is discerned by computing the overlap among the features covered by the patterns involved.

* **Step2.3.1.** If all the features needed by the requirement are fulfilled and there is not overlap between features provided by the patterns (features not repeated), then the patterns may be combined (case ii).
* **Step2.3.2.** If there is a significant overlap between features, then just one pattern should be selected (case iii).

The details of the algorithm are shown in Fig. 6. The different cases mentioned above are highlighted by means of comments. The output of the algorithm for each \( RQ \) is either a strong suggestion for one or more patterns to be used or a subset of candidate patterns among which the analyst must take a decision.

The running example is used now to illustrate the behavior of the algorithm. Examples of the three different cases explained above will be highlighted again. Table 8 shows the results obtained for

### Table 5
Binary matrix relating patterns and RIA features (ptm).

<table>
<thead>
<tr>
<th></th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
<th>F6</th>
<th>F7</th>
<th>F8</th>
<th>F9</th>
<th>F10</th>
<th>F11</th>
<th>F12</th>
<th>F13</th>
<th>F14</th>
<th>F15</th>
<th>F16</th>
<th>F17</th>
<th>F18</th>
<th>F19</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>P6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>P7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>P8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>P9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

### Table 6
Correspondence matrix relating requirements and RIA patterns.

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ1</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>G</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>RQ2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>RQ3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>RQ4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>RQ5</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>RQ6</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>RQ7</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

### Table 7
Combinations of DRR and DPR.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0 &lt; DPR &lt; 1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ ( \cap P = \emptyset )</td>
<td>Unfeasible</td>
<td>Unfeasible</td>
</tr>
<tr>
<td>0 &lt; DRR &lt; 1</td>
<td>Unfeasible</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>P ( \subseteq RQ )</td>
<td></td>
<td>P = RQ</td>
</tr>
<tr>
<td>1</td>
<td>Unfeasible</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RQ ( \cap P \neq \emptyset )</td>
<td>Unfeasible</td>
<td>P = RQ</td>
</tr>
<tr>
<td></td>
<td>RQ ( \subseteq P )</td>
<td>Unfeasible</td>
<td>P = RQ</td>
</tr>
</tbody>
</table>
the metrics calculated. Shadowed cells in Table 8 emphasize those patterns that were considered as candidates for a requirement (RIA Patterns Elicted in Fig. 1).

An example of case (i) (identified in step 2.1 in the description of the algorithm) is the case of P6 and P7 that obtain values of 1 in DPR and DRR for RQ3 and RQ4, respectively. As an example of case (ii), consider now RQ2 that presents values of 1 in DPR for two patterns, P2 (Data Event Notification) and P5 (Partial page refresh). However, the values obtained in DRR for P2 and P5 are different from 1 (0.455 and 0.545, respectively). These results show evidences that the combination of the patterns may cover the whole set of RIA features required by the requirement. However, Table 8 does not show if they cover the same features or not. This fact can be detected by adding up P2 and P5 rows (step 2.3 in the algorithm) in ptm (Table 9). A value of 2 in a cell means that there is an overlap. In this case there not exists overlap due to all the cells having the value of 1 and the resulting row is equal to the row for RQ2 in rtm. So, this would be an example of case (ii) (identified in step 2.3.1 in the algorithm) where both patterns must be used. If we consider now RQ1 one can infer that P3 and P4 are the candidates. If we add up both rows in ptm (Table 10), then some overlaps may be detected. The problem for the analyst is to decide whether the
right solution is combining the patterns (case ii) or discarding one of them (case iii). In this case, since most of the features are overlapped, the developer should select just one of the patterns (case iii identified in step 2.3.2 in algorithm). Note that both patterns differ only in the communication schema used (dataset when the client confirms the shopping cart or tuple when a client prebooks a seat). Precisely these are two of the scenarios identified in Section 2.1, so both of them may be applied (P3 for the first scenario and P4 for the second one). We have arrived at this situation due to RQ1 being coarse grained. If we had chosen to divide RQ1 into two different requirements then the situation would have been different (case i). So there may be a trade-off between the granularity of the requirement and the possibilities offered by the approach, as discussed in Section 4.

Finally, there are two different patterns that fit RQ7, Module Tabs (P8) and Left Navigation Menu (P9). Thus, the two patterns could be indistinctively used to address the requirement (case iii identified in step 2.2 in algorithm). In this situation, a deeper analysis of the patterns and the requirement may help to discern whether there is a pattern more adequate than the other ones. As an example, the Module Tabs is recommended for a navigation through a number of items smaller than 8 or 10 and short names for these items while the Left Navigation Menu one is used when the number of items may be higher and there is no restriction on the names length. In our running system, pattern P8 was selected since our menu items were not affected by the constraints.

For the sake of simplicity, the rest of patterns analyzed have not been shown in the process and this is why no patterns were suggested for RQ5 and RQ6. However, the results obtained in the whole analysis showed evidences that Field content autocompletion (Brambilla et al., 2010) and Calendar picker (Yahoo!-Patterns, 2011) patterns could be used for dealing with RQ5 and RQ6, respectively, not shown in ptm for simplicity.

Once the RIA patterns are identified, they are introduced into the system by using the model weaving approach (process d in Fig. 1) described in Rodríguez-Echeverría et al. (2010). This process merges the model of the legacy system (obtained in step a in Fig. 1) with the pattern models by using Aspect-Oriented Modeling (AOM) principles (Clarke and Baniasad, 2005). The whole weaving process is not shown here since it was already described in Rodríguez-Echeverría et al. (2010). However, in order to make the paper self-contained, the case study used in Rodríguez-Echeverría et al. (2010) (the tickets system) has been also used in this work so that the reader may refer to Rodríguez-Echeverría et al. (2010) to observe the models resulting after weaving some of the patterns into this system.

4. Evaluation and discussion

The case study provided in this paper is not casual because it covers a common set of requirements that recurrently appears in the development of Web applications. However, in order to evaluate the approach, MIGRARIA has been also applied to several legacy systems with RIA modernization requirements. Two different kinds of applications have been used. On the one hand, some demo systems with the aim of illustrating the extension and applicability of the approach. This first analysis was conceived to confirm two different hypotheses:

- H1 – The set of RIA features selected are representative enough for common modernization scenarios.
- H2 – The set of patterns defined in ptm are suitable to face common modernization scenarios.
On the other hand, a set of real applications developed by an external company were used to evaluate whether the results obtained in the three demo systems could be also extrapolated to bigger systems. Particularly, the objective of this second analysis was to confirm the next hypotheses:

- **H3** – The measures presented in Section 3.3 provide useful results and they may be used to tune up the computations performed in step 3.2 of the algorithm.
- **H4** – Results suggested by the approach are coherent with those provided by RIA developers.

### 4.1. Demo systems

The demo systems evaluated include the ticketing system shown throughout the paper together with a *University Agenda* and a *Conference Review System (CRS)* both described in *Migraria-Project (2012)*. These systems were selected since they cover all the RIA features considered in the approach. Although a whole description of the case studies may be found in *Migraria-Project (2012)*, the modernization proposed for these systems may be summarized as follows. The *University Agenda* system aims at providing a shared agenda for faculty members including students, professors and administrators. This system is a typical example of data-driven Web application with many CRUD operations over the underlying information system. The system was modernized by providing (i) synchronization issues, such as offline access to the teaching schedules or client to client notifications and messaging; (ii) improving the user experience with the system by adding functionalities like searching for professors by text suggestions or drag-and-drop-based date modifications.

The main goal of CRS is to manage paper reviewing process for a conference where three different user roles are defined: (i) authors, who submit papers to the conference; (ii) PC members, who are responsible for reviewing the papers submitted by authors; (iii) PC chair, who assigns papers to review to the PC members and, finally, decides the papers to be accepted for the conference. The modernization proposed for this system focuses on improving the interaction within the system (by adding new RIA widgets and components) but also on introducing new synchronization functionalities such as disconnected work (allowing to write offline reviews that are synchronized as soon as user reconnects) or server pushing notifications (notifying that co-reviewers’ new comments are available in the reviewers’ discussion for a paper).

Based on the application of the approach to the demo systems, we observed that the set of RIA features considered was representative enough to describe all the new RIA requirements introduced by the different modernizations performed. These results come to confirm and validate the set of RIA features as an important contribution of the approach being enough to cover the most common modernization scenarios (*hypothesis H1*). Although the selection of the RIA features for every RQ may imply some RIA expertise, the mere existence of such a set of RIA features help at reducing the effort of proper RQ description in terms of RIA concern, by providing a common and unique reference. Another interesting result signals that the *Partial refreshment* feature may provide redundant information. It was identified as involved in every presentation-related requirement and pattern. This suggests that this feature could be considered as an intrinsic characteristic of RIA applications and, maybe, it could be removed from our RIA tables (implicitly included in RIA presentation issues). Additionally, we verified that the selection of RIA features for each pattern provides the engineer with a representative description since the patterns could be applied for the modernization scenarios proposed.

Regarding the suitability of the patterns, we observed that the set of patterns considered in the case study was also representative enough for covering the RIA modernization scenarios proposed by these three demo systems (*hypothesis H2*). However, being aware of the influence of such controlled scenarios in the results, we evaluated the approach also with a set of real projects to check whether the results were also applicable to these systems.

### 4.2. Real systems

This section shows the application of the approach to some real projects developed by an external company, Homeria Open Solutions. Homeria is a company with more than fifty RIA projects tackled, so that their developers have a strong expertise in RIA technologies. These projects were developed by using the RUX method and its case tool (RUX-Tool) (*Linaje et al., 2007*) that works integrated in Webratio (the WebML CASE tool), the same tools used to modernize Homeria applications. The company allowed us to access to four systems representative of all the RIA projects developed. These projects were classified according to their application domain: education, e-commerce, tourism, and corporate. Concretely, the projects evaluated were: (i) Educatec Campus (education), a virtual campus for the management of teaching courses; (ii) Demostore (e-commerce), an e-shop system that the company clients may configure for their own businesses; (iii) Tourism Information System (tourism) for the city of Cáceres; and (iv) Homeria Website (corporate). We applied the approach taking as input the requirements of these five projects and checked whether the patterns suggested were used in their development. Note that these projects were selected since they were representative for different domains but also for different RIA functionalities. Synchronization issues were the most important concerns in Educatec Campus, due to it having a huge number of users that may work in offline mode or been notified by server pushing mechanisms. Presentation concerns were of utmost importance for Tourism Information System or Homeria corporate Web, since their main goal was to gain client attention by using a very attractive user interaction experience. In the Demostore, synchronization and presentation issues were both important due mainly to the synchronizations required by the shopping cart and the need for an attractive user experience (to keep buyers in the shop), respectively.

*Table 11* presents the information for the projects analyzed. This table shows the URL for each project, its category, the patterns identified by means of our approach, the patterns selected by Homeria developers and percentage of patterns both identified by the approach and used in the development (true positive rate). In order to simplify the comparison between the sets of patterns presented, the name of each pattern has been replaced by a corresponding code. The correspondence between these codes and the patterns is shown in *Table 12*. Finally, the patterns identified by the approach but not selected by the developers (false positives) and those used by the developers but not identified by the approach (false negatives) were highlighted in bold in the corresponding columns, respectively.

Based on these data, we extracted some conclusions that are explained next. Firstly, we observed that the percentage of patterns that were identified by both the approach and developers (true positive rate) was higher than 55% in all cases. These data come to confirm that the approach obtains, in general, results coherent with those obtained by real developers (*hypothesis H4*). Note that, in the worse case (*Cáceres Tourism system*), more than the half of the patterns would have been automatically suggested by our approach implying an important reduction in development time.

---

4 [www.homeria.com.](http://www.homeria.com)

5 The capital of the same name province, in the autonomous community of Extremadura, Spain.
Secondly, we observed that the synchronization patterns suggested by the approach and used by developers are the same in all the systems (e.g., patterns P1, P3 and P17 in Educatec system or P1 and P2 in Homeria corporate). This shows evidences that the limited number of synchronization patterns considered in our approach was representative enough to cover all the appearing situations of data transferring between server and client (and vice versa). In that sense, we observed that a proper combination of these patterns could solve the requirements related with the data synchronization for each system (related with hypothesis H2). It is worth to take into account this fact since most of the modernization processes require synchronization matters.

Regarding presentation issues, we observed that most of the false positives and negatives obtained in the results are related to these patterns. This is mainly due to the wider variety of patterns for presentation issues that causes that patterns suggested by the approach may be realized by different RIA widgets or designs. As an example, while the approach suggested pattern P12 (Dynamic tooltips on page data) in Cáceres Tourism, developers used the union of pattern P14 (Tooltip invitation) and P5 (Cursor invitation) to fulfill the requirements covered by pattern P12. This is a clear example where different patterns may be used to implement the same functionality. We also observed that, sometimes, these results might be influenced by the RIA expertise of the developers. For instance, we realized that developers in Homeria recurrently utilized the same patterns for solving similar but different situations. This is due to the fact that developers, in general, often use well-known designs or widgets even when unknown alternative designs (sometimes even better) may be available. Nevertheless, analysts in Homeria observed that these patterns (the false positives obtained by the approach) could be also used in their designs in order to obtain the same functionalities implemented in the systems. This shows that the approach may potentially find out new patterns that developers usually not consider mainly due to pattern existing ignorance or lack of use.

Regarding false negatives, we observed situations where developers used patterns that the approach does not suggest (neither other patterns that substitute them). Examples of these situations are patterns P7 and P8 in Educatec or P5 in Demostore. In these cases, the approach could be also adapted to include other patterns suggested by developers or to weight patterns so that those mostly used by the company are prioritized and more likely suggested. This is the way in which the algorithm can be tuned up introducing some kind of subjectivity if needed. In other words, the designer always may move back and review the requirements granularity level used in rtm (hypothesis H3).

Anyway, despite the variety of presentation patterns that may be selected, we observed that those selected by the approach and also used by developers (true positives) always appear as true positives in all the systems where are included. Of course, this situation may be also influenced by the definition of the requirements related to these patterns, which may be similar in all the systems. However, this is a good indicator since those requirements are quite recurrent in Web systems, e.g. having a widget with content-based navigation (e.g. Carousel).

Finally, our approach was tested by different sets of students of a Web Engineering course at Universidad de Extremadura in order to evaluate whether the approach is also useful for non-professional RIA developers. The analysis involved three groups of students with different level of RIA knowledge and each group applied the approach to different systems where we had also previously applied the approach. We observed that, independently of the system modernized, the true positive rate increases accordingly to the RIA expertise level of the different groups. Analogously, we observed that the true positive rates obtained with students were, in general, slightly worse than those obtained with Homeria RIA developers. Based on these results, we concluded that the approach is aligned with the way of selecting patterns by professional RIA developers (hypothesis H4). Additionally, a group of students performed the modernization of the same demo systems without the support of the approach. The results obtained by this group were worse than those obtained by all the groups that used the approach. This fact shows that the conceptual framework provided by the rtm template is helpful to assist the Web analyst to properly elicit modernization requirements conveying RIA features.

As a summary, the results described in this section show evidences that the metrics utilized in the framework works properly (hypothesis H3) since they mostly provide results coherent to the identified by developers (hypothesis H4) with a better correlation when RIA experts are considered.

### 5. Related works

This section covers different areas related to our process: first, works focused on Web requirements (specially RIA-related) and the derivation of architectural design from them and, second, works dealing with design patterns for modeling RIA.

### Table 11
 Systems developed by Homeria and used to evaluate the approach.

<table>
<thead>
<tr>
<th>System</th>
<th>URL</th>
<th>Category</th>
<th>Patterns identified</th>
<th>Patterns used in development</th>
<th>% matching patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Educatec Campus</td>
<td><a href="http://campus.educatec.buz/">http://campus.educatec.buz/</a></td>
<td>Education</td>
<td>P1, P3, P4, P17, P6</td>
<td>P1, P3, P4, P17, P5, P7, P8</td>
<td>57.1 %</td>
</tr>
<tr>
<td>Demostore</td>
<td><a href="http://demostore.homeria.com/">http://demostore.homeria.com/</a></td>
<td>E-commerce</td>
<td>P2, P3, P4, P9</td>
<td>P2, P3, P4, P9, P5</td>
<td>80 %</td>
</tr>
<tr>
<td>Cáceres Tourism</td>
<td><a href="http://www.turismocaceres.org/">http://www.turismocaceres.org/</a></td>
<td>Tourism</td>
<td>P10, P11, P4, P3, P12</td>
<td>P10, P11, P4, P3, P13, P14, P5, P7</td>
<td>55.55 %</td>
</tr>
<tr>
<td>Homeria Corporate</td>
<td><a href="http://www.homeria.com/">www.homeria.com/</a></td>
<td>Corporate</td>
<td>P1, P2, P3, P10, P15</td>
<td>P1, P2, P3, P9, P10, P13, P16, P5</td>
<td>75 %</td>
</tr>
</tbody>
</table>

### Table 12
 List of patterns identified in the projects.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Data Event Notification</td>
</tr>
<tr>
<td>P2</td>
<td>Bulk data transfer replication</td>
</tr>
<tr>
<td>P3</td>
<td>Carousel</td>
</tr>
<tr>
<td>P4</td>
<td>Partial page refresh</td>
</tr>
<tr>
<td>P5</td>
<td>Search pagination</td>
</tr>
<tr>
<td>P6</td>
<td>Sign-in continuity</td>
</tr>
<tr>
<td>P7</td>
<td>Cursor invitation</td>
</tr>
<tr>
<td>P8</td>
<td>Reflektor</td>
</tr>
<tr>
<td>P9</td>
<td>Top navigation bar</td>
</tr>
<tr>
<td>P10</td>
<td>Calendar picker</td>
</tr>
<tr>
<td>P11</td>
<td>Autocompletion</td>
</tr>
<tr>
<td>P12</td>
<td>Dynamics tooltips on page data</td>
</tr>
<tr>
<td>P13</td>
<td>Page grids</td>
</tr>
<tr>
<td>P14</td>
<td>Tooltip invitation</td>
</tr>
<tr>
<td>P15</td>
<td>Data filtering</td>
</tr>
<tr>
<td>P16</td>
<td>Accordion</td>
</tr>
<tr>
<td>P17</td>
<td>Client reconnection</td>
</tr>
</tbody>
</table>
Regarding Web requirements and design derivation, the need for incorporating requirements engineering techniques into the Web engineering domain has been widely highlighted in works like Aguilar et al. (2010), Escalona and Koch (2004) where surveys of existing methodologies for capturing Web requirements are presented. In this context, Koch et al. (2006) presents an approach to derive design models from Web requirements based on models transformations. However, neither the surveys nor the work in Koch et al. (2006) deals with RIA features. The work in Luna et al. (2010) introduces an analysis of the new requirements appearing due to RIA capabilities and extends the meta-model for specifying Web requirements introduced in Escalona and Koch (2006) to cover these RIA features. More recently, in Fraternali et al. (2010) some RIA capabilities are described to be captured in WebML. However, both Luna et al. (2010) and Fraternali et al. (2010) lack of a process to derive design artifacts from requirements.

Regarding design patterns, their need in Web engineering has been widely highlighted in the literature (Brambilla et al., 2010; Koch et al., 2009; Rodríguez-Echeverría et al., 2010; Yahoo!-Patterns, 2011; Vora, 2009). Early works (Rossi and Koch, 2002; Rossi et al., 2000), proposed personalization patterns to adapt the presentation, content and navigation of a page according to the user's preferences. However, these works just considered traditional Web 1.0 systems where RIA features are not included. One of the most popular pattern collections including RIA features is the Yahoo catalog (Yahoo!-Patterns, 2011). These patterns deal with recurrent presentation and navigational issues in RIA developments. A similar catalog of RIA patterns is described in Vora (2009), however, it just presents textual descriptions and the pattern models are not described.

In Brambilla et al. (2010), the authors introduce some RIA patterns modeled in WebML (Ceri et al., 2002). The novelty of this approach relies on the integration of the patterns into WebRatio, the supporting tool for WebML, so that the code for these patterns may be automatically generated in AJAX.

In Koch et al. (2009), the authors introduce a way to model RIA patterns at a higher abstraction level independently from the Web methodology. They focus on the abstract presentation of some RIA features that are modeled by state machines so that the later integration of these diagrams with existing Web methods could be achieved by model transformations, as authors suggest. In that sense, our approach suggests RIA patterns from the system requirements, covering the gap between requirements and architectural design.

Unlike the aforementioned approaches (focused on presentation concerns), the work in Rodríguez-Echeverría et al. (2010) proposes patterns to deal with communication and synchronization issues. However, this work does not deal with the systematic identification and selection of the most suitable RIA patterns.

In Hasheminejad and Jalili (2012), the authors present an approach to automatically suggest design pattern(s) to the developer according to a given design problem. The process consists of two steps Learning Design Patterns and Design Pattern Retrieval. In the former, the authors take as input the design patterns (e.g. Gamma et al., 1995) with the goal of learning knowledge from experts (e.g. a particular design label for each given design problem). In the latter a candidate pattern type similar to a given problem is determined based on a text classification approach. To this purpose, the authors define a vector based on each problem design characteristics. Similarly, we propose a matrix where each requirement is analyzed based on RIA characteristics. However, while the approach in Hasheminejad and Jalili (2012) is defined to be applied during a traditional analysis phase, where the whole system specification is available, we aim at proposing a modernization process where only the new modernization requirements are available. Moreover, the approach in Hasheminejad and Jalili (2012) was evaluated by using traditional design pattern catalogs and not with RIA patterns.

6. Conclusions

This paper has presented part of a process to modernize legacy non model-based Web applications by adding RIA capabilities.

The process is based on the utilization of traceability matrices that align requirements and RIA patterns with the RIA features involved in them. The process provides a final traceability matrix where the most suitable RIA patterns for each new requirement are suggested. The values obtained in this matrix are normalized by using two different measures, i.e. Degree of requirement realization (DBR) and Degree of pattern realization (DPR). These measures are used by an algorithm that selects the most suitable patterns for each requirement. We applied the process to a running example and illustrated how the developer may find situations where a pattern perfectly matches one requirement, the combination of several patterns is suitable for one requirement or there are several patterns that could be applied to realize one particular requirement. Finally, once the patterns are selected, they are weaved into the legacy models so that those RIA functionalities implemented by the patterns are incorporated into the system. For the sake of simplicity, we just illustrated the patterns that were related with our running example; however, we performed an evaluation process by applying the approach to other systems where we checked the correctness of the features and patterns used and evaluated the metrics and algorithm to select candidate patterns. The approach has been implemented using simple datasheets that allows playing with the matrices and performing a kind of sensitive analysis. The initial catalog of patterns that has been used for validation purposes is based on Brambilla et al. (2010), Rodríguez-Echeverría et al. (2010), Vora (2009), Yahoo!-Patterns (2011).

We state that based on our approach, the reusability of the patterns is clearly improved since some of the artifacts such as the patterns traceability matrix is built once and used in any modernization process. Moreover, the modernization process proposed may be easily integrated with different Web methodologies where patterns for RIA features may be modeled.

Although the application to legacy model-based Web applications has been illustrated, the process is part of a larger project, MIGRARIA, where the starting point is a legacy non model-based Web application. The proposed approach may be also useful to build new RIAs by deriving design patterns from requirements, since the main contribution of the presented work is covering the gap between requirements elicitation and architectural design in the development of RIAs.

As future work, we plan to analyze the utilization of the information contained in the traceability matrices to keep forward and backward traceability among requirements and architectural design artifacts. Additionally, the adoption of this approach in parts of some modernization case studies is giving us some keys to enhance our conceptual framework to assist the analyst in aligning requirements and RIA features, thus minimizing the expertise demanded.

Acknowledgments

This work has been funded by Ministerio de Economía y Competitividad under grant TIN2011-27340 and by Gobierno de Extremadura.

References


