A Goal-Oriented Requirements Engineering Approach to Distribute Functionality in RIAs

José Alfonso Aguilar[1,3], Sven Casteleyn[2], Irene Garrigós[1], and Jose-Norberto Mazón[1]

[1]Department of Software and Computing Systems
University of Alicante, Spain
[2]Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València, Spain
[3]Computer Science Faculty
University of Sinaloa, Mexico
{ja.aguilar,igarrigos,jnmazon}@dlsi.ua.es
sven.casteleyn@upv.es

Abstract. The challenges and complexity involved in the design and development of Rich Internet Applications are well recognized, and most existing Web 1.0 design methodologies have been augmented to tackle these. However, the focus is mostly on technological and engineering challenges, while only providing limited support for the requirements analysis. This paper addresses this hiatus by extending an existing goal-oriented Web requirements analysis approach to support RIAs. Specifically, we present an approach (based on the Pareto efficiency) to evaluate and select desirable client/server distribution of functionality for RIAs, and guide the designer towards an optimal solution that provides a balanced maximization of the non-functional requirements. We illustrate our approach by using an excerpt of a real-world case study of a bioinformatics company.

Keywords: Web Engineering, Rich Internet Applications, Goal-Oriented Requirements Analysis

1 Introduction

Due to the Web's highly heterogeneous user base and its constant evolution, requirements specification is an important yet difficult task in the Web engineering field. Importantly, a key factor in a Web application's success is to ensure that the functionality required by the users is fulfilled, while at the time facilitating their browsing session [10]. This is tightly related to both (i) functional requirements (FRs) that describe the system services, behavior or functions, e.g., "download a sales report"; and (ii) non-functional requirements (NFRs) which specify a constraint over that functionality, e.g., "usability" [17]. Therefore, Web methodologies should be able to provide a requirements analysis phase in which functional and non-functional requirements are considered.

Unfortunately, both in industrial cases and in Web engineering methodologies, requirements are not sufficiently taken into account when developing a Web application [3]. For stakeholders[1], this results in a mismatch between their expectations and the resulting Web application, and for users in a Web application that poorly supports their requirements and does not fulfill their needs.

As a natural evolution of the Web, Rich Internet Applications (RIAs) enable client-side logic, offer better responsiveness and interactivity, and provide more elaborate and attractive presentation. Their main goal is to tackle the limitation imposed by traditional Web 1.0 technology [6], and provide the user with a richer experience, similar to that of desktop applications. While offering more possibilities to satisfy the user's expectations, RIAs are also considerably more complicated to design and develop. Existing Web design methodologies have been extended to support RIAs, but the proposed extensions mainly focus on presentation issues or interaction capabilities [13]. Requirements analysis is largely left to the discretion of the designer and no systematic, tailored support for RIAs is provided. Nevertheless, this is not a trivial task, as the designer needs to balance classical Web requirements, such as those functionalities (i.e., FR) related to navigation, and RIA specific NFR, such as "Responsiveness" or "Bandwidth reduction". Furthermore, the designer needs to take into account that the distribution of functionality between client and server influences the satisfaction of the NFRs; the choice where to place certain functionality is thus critical.

In our previous work [2, 11], we presented a GORE (Goal-Oriented Requirements Engineering) approach based on the *i** modeling framework to specify Web requirements. In this paper we extend that approach to support requirements engineering in RIAs. Specifically, we focus on studying the requirements distribution between client and server in order to provide a balanced maximization of the satisfaction of NFRs. For this purpose, we present a proposal that allows the designer to obtain and evaluate good client/server configurations (requirements distribution), i.e. those that maximize the NFRs. Our proposal is based on the Pareto efficiency, which is useful when there are multiple competing and conflicting objectives [5] that need to be balanced, as is the case here. In essence, a group of non-dominated optimal solutions (called Pareto front) is created, i.e., solutions that optimize the NFRs (corresponding to *softgoals* in our GORE framework) in such a way that no single NFR can still be improved, without negatively affecting another. From these balanced optimizations, the designer can select the final solution by taking into account the importance of the different NFRs. We present our results by using an excerpt of a real-world case study of a start-up bioinformatics company in order to effectively show how our approach can aid designers to obtain an adequate client/server distribution of functionality in RIAs.

This paper is structured as follows: Section 2 presents the related work. Section 3 describes our previous work on GORE for the Web 1.0. The contribution

[1] Individuals or organizations who affect or are affected directly or indirectly by the development project in a positive or negative form [17].

of this work is presented in Section 4. Section 5 describes a real world case study. Section 6 presents the ongoing work related to the implementation of our approach. Finally, the conclusion and future work are presented in Section 7.

2 Related Work

Research in Web engineering has shown that RIA development is a difficult challenge which requires extending the traditional Web engineering approaches [14], i.e., to improve the development process and adapt the conceptual models in order to support richer presentation and interactive user interfaces.

Several Web 1.0 Web engineering approaches have been extended to support RIAs, i.e., (i) **WebML** (Web Modeling Language) [4], (ii) **OOWS** (Object-Oriented Approach for Web Solutions Modeling) [19], (iii) **OOHDM** (Object-Oriented Hypermedia Design Method) [18], and (iv) **UWE** (UML-based Web Engineering), which has three extensions for RIA development (UWE-Patterns [9], UWE-R [12] and UWE-RUX [15]). For a complete analysis of the techniques used for requirements analysis by the approaches described in this paragraph we refer the reader to [3].

Some extension have been created in order to be combined with Web engineering approaches as the ones aforementioned, these are: (i) the **RUX-Model** [16], solely for the design of UIs for RIAs over existing HTML-based Web Applications in order to give them multimedia support, and (ii) the **OOH4RIA** [13], which defines a model-driven development process to obtain a complete RIA for the GWT (Google Web Toolkit) framework, and is supported by a CASE tool.

On the other hand, new methods have been created exclusively for the development of RIAs. **IAML** (Internet Application Modelling Language) [21] is a modelling language that directly uses RIA concepts as first-class modelling citizens [20]. IAML has a CASE tool for the development of RIAs based on EMF (Eclipse Modeling Framework) and GMF (Graphical Modeling Framework). The **ADRIA** (Abstract Design of Rich Internet Applications) [7] method is an UML-based approach, whose design activities depart from an object-oriented analysis, and focus on the design of events triggered by user interactions.

In some of these approaches, e.g. OOH4RIA, IAML, RUX, the requirements engineering phase is not considered at all. In others, classical software engineering techniques are used, such as use cases in UWE, WebML, OOWS, OOHDM and ADRIA, activity diagrams in UWE, WebML and OOHDM, or task models in OOWS. These techniques are inherited from their respected Web 1.0 methods, and are not well-adapted to take RIA's requirements into account, or deal with RIA specificities (i.e., client/server distribution). Our approach complements these techniques, is method-independent, and allows to focus on satisfying user goals and objectives by means of our goal-oriented approach, while taking specificities of RIAs, in particular the client/server distribution, into account.

Finally, some interesting goal-oriented techniques for requirements analysis have been proposed for non-Web engineering approaches, i.e., [8]. This work evaluates *i** models based upon an analysis question (what-if analysis) and the human judgment. To this aim, this procedure uses a set of evaluation labels to represent the satisfaction or denial level of each element in the *i** model. Unfortunately, these general approaches have not been adapted to Web engineering.

3 Specifying Requirements in Web Engineering

In this section we summarize our requirements analysis approach for Web 1.0 applications based on the *i** modeling framework [1, 2, 11]. This approach is the starting point of our approach for analyzing the client/server distribution of functionality when RIA-specific requirements are considered.

The development of Web applications involves different kinds of stakeholders with different needs, which depend on each other to achieve their goals, perform several tasks or obtain some resources. For example, the Web administrator relies on new clients for obtaining data in order to create new accounts. The *i** framework [22] is a GORE framework used to explicitly analyze and model these relationships among multiple stakeholders (actors in the *i** notation), along with their goals, tasks and resources. Essentially, the *i** framework consists of two models: the strategic dependency (SD) model, to describe the dependency relationships (represented as ⊸D⊸) among various actors in an organizational context, and the strategic rationale (SR) model, used to describe actor goals and interests and how they might be achieved. Therefore, the SR model (represented as a dashed circle ◌) provides a detailed way of modeling the intentions of each actor (represented as a circle ○), i.e., internal intentional elements and their relationships:

- A goal (elipse ⬭) represents an (intentional) desire of an actor. Interestingly, goals provide a rationale for requirements but they are not enough for describing how the goal will be satisfied. This can be described through means-end links (⊸▷) representing alternative ways for fulfilling goals.
- A task (hexagon ⬡) describes some work to be performed in a particular way. Decomposition links (—+) are useful for representing the necessary intentional elements for a task to be performed.
- A resource (rectangle ▭) represents some physical or informational entity required for the actor.
- A softgoal (eight-shape ∞) is a goal whose satisfaction criteria is not clear-cut. How an intentional element contributes to the satisfaction or fulfillment of a softgoal is determined via contribution links ($\xrightarrow[\text{hurt}]{\text{help}}$). Possible labels for a contribution link are "make", "some+", "help", "hurt", "some-", "break", "unknown", indicating the (positive, negative or unknown) strength of the contribution.

Our previous work [2] presented an approach for adapting the *i** framework for specifying requirements typically encountered in Web applications (this adaptation is supported by a metamodel described in Fig. 1):

- Service requirements refer to the internal functionality the system should provide to its users, e.g., "register a new client".

– Navigational requirements aims to define the paths needed for the user to browse the Web applications, e.g., “consult products by category”.
– Layout requirements for defining the visual interface for the users, e.g., “present certain text style for headers”.
– Personalization requirements allow the designer to specify the desired adaptation functionality to be performed in the final Web application, e.g., “adapt font for visual impaired users”.
– Content requirements define the content that the Web application presents to its users, e.g., “product information”.
– Non-Functional requirements are related to quality criteria that the intended Web system should achieve and that can be affected by other requirements, e.g., “responsiveness”.

An example of an *i** model for Web engineering can be seen in Fig. 3 with explanation in Sect. 5.

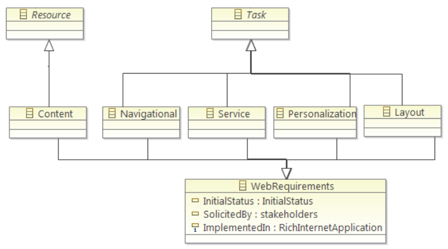

Resource
Task
Content
Navigational
Service
Personalization
Layout
WebRequirements
InitialStatus : InitialStatus
SolicitedBy : stakeholders
ImplementedIn : RichInternetApplication

Fig. 1: An excerpt of the *i** requirements metamodel for Web domain.

Finally, it is worth noting that FRs (navigational, layout, service and personalization) are represented as tasks, while the requirements related to the data (content) are represented as resources. There is thus a clear separation between these two concerns; in this paper we focus on the (distribution of) FRs. Also, NFRs are represented as softgoals. For further details, we refer the reader to [2].

4 Optimizing RIA functionality distribution by using Pareto efficiency

A critical task in RIA requirements engineering is to determine a *good* distribution of functionality (client/server). In other words, for each FR the designer needs to decide where it will contribute best to the realization of the NFR or softgoals (according to our framework), i.e. on the client or on the server side. This is not a trivial decision, as each requirement influences different softgoals differently depending on whether it resides on the server or the client. In addition, maximizing one softgoal often leads to decreased support for another softgoal, so there is no notion of a single best client/server configuration and a tradeoff among softgoals should be made.

Pareto efficiency [5] is an ideal candidate to tackle this problem, as it is designed to balance competing concerns. It is a notion from economics widely applied to engineering, which is described as follows: given a set of alternative allocations (server or client) and a set of individuals (requirements), allocation A is an improvement over allocation B only if A can make at least one individual better (maximizing softgoals) than B, without making any other worse (weakening softgoals). Therefore, in our approach, a Pareto optimal (client/server) configuration is one so that no other configuration better satisfies one single softgoal, while satisfying the others equally. The set of Pareto optimal configurations are thus candidate *good* distributions. It includes the solutions that best satisfy each single softgoal, but also those that provide an optimal tradeoff between softgoals. As such, it forms a perfect basis to make a well-informed design decision for functionality client/server distribution.

More formally, finding the set of Pareto optimal configuration can be defined as the problem of finding a (decision) vector of decision variables X (i.e., a valid client/server distribution), which maximizes a vector of M objective functions $f_i(X)$, (i.e., the satisfaction of softgoal i in client/server distribution X, where $i \in \{1, ..., M\}$ (with M the amount of softgoals). To do so, the concept of domination between vectors is defined as follows: a decision vector X is said to dominate a decision vector Y $(X \succ Y)$ if and only if their corresponding objective vectors of objective functions $f_i(X)$ and $f_j(X)$ satisfy: $\forall i \in \{1...M\} : f_i(X) \geq f_i(Y)$ and $\exists i \in \{1, ..., M\} : f_i(X) > f_i(Y)$. We then say that all decision vectors that are not dominated by any other decision vectors form the Pareto optimal set, while the corresponding objective vectors are said to form the Pareto front.

Following the Pareto efficiency, we defined the following steps to determine the Pareto optimal configuration for distributing FRs between client and server, while the softgoals are balanced and maximized (Figure 2). Our approach supports bith scenarios: (i) the evolution of an existing Web 1.0 specification to support RIAs, and (ii) from-scratch specification.

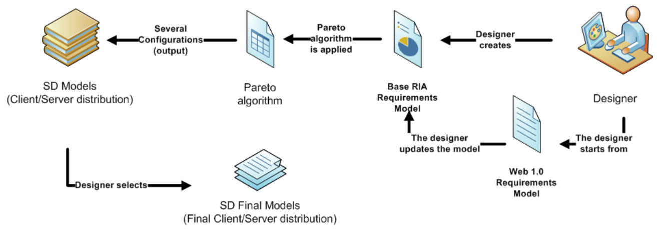

Several Configurations (output)
Pareto algorithm is applied
Designer creates
SD Models (Client/Server distribution)
Pareto algorithm
Base RIA Requirements Model
Designer
The designer updates the model
The designer starts from
Web 1.0 Requirements Model
Designer selects
SD Final Models (Final Client/Server distribution)

Fig. 2: An overview of our RIA requirement analysis approach.

Before defining each step of our approach for dealing with the distribution of functionality, it is worth noting that the distribution of data is also a relevant requirement when developing RIAs, and in fact, both are coupled. Indeed, when deciding to implement particular functionality on the client, the relevant data needed to realize this functionality needs to be duplicated on the client. A straightforward solution is thus to duplicate in the client any resource (content requirement) that is related to a task (navigation, layout, personalization or service requirement) located in the client. We are currently applying this solution, allowing us to focus only on functionality. However, it needs to be mentioned that the duplication of data and its communication to the client causes some additional effects, which might influence, both positively and negatively, the maximization of softgoals. For example, duplication and communication of data to the client side might cause slower initialization, but on the other hand, re-use of this data for different tasks will increase responsiveness. We are therefore considering in our future work to also adopt content requirements in our Pareto algorithm to include their effect on the maximization of softgoals.

Step 1.- Create the base RIA requirements model. When no Web 1.0 artifacts are available, a Web 1.0 requirements model is created (see Sect. 3), but additionally including RIA specific requirements, softgoals and the contribution links they induce. The RIA specific requirements are those that are only realizable using RIA technology (e.g., "Provide a graphical view", "Capture webcam"). The RIA specific softgoals arise from the fact that RIA technologies are used; they would otherwise not be present, possible or relevant. Examples include "Responsiveness" or "Low bandwidth". At this point, we do not yet take into account the distribution of FRs among client and server. Hence, the label of the contribution links (help, hurt, some+, some-, make and break) cannot yet be determined, as this is dependent on where each requirement is implemented. For example, the requirement "Browse result set" will "help" the softgoal "Responsiveness" if implemented on the client, but "hurt" it when implemented on the server. Therefore, we leave the label of the contribution links as "unknown" at this moment; later, when we take into account server/client distribution, we will assign the correct label to each of the contribution links. In some cases, the label of the contribution can be directly assigned[2]: if the contribution involves a requirement that can only be realized on server or client side (i.e., "Provide graphical view" needs client-side technology and is thus only realizable on the client side) or for softgoals to whom a contribution is client or server independent (i.e., "Provide proprietary format" always "hurts" "Compatibility" as it concerns a non-standard format).

In case a Web 1.0 requirement model was already available, the model is evolved as follows: the RIA specific requirements, softgoals and contribution links (with "unknown" label) are added to the existing model. For contributions that were already present in the Web 1.0 requirement model, their label is replaced by "unknown" if it may be influenced by the client/server distribution of the requirements.

Step 2.- Obtain the Pareto optimal client/server distribution. We provide a set of sub-steps based on the Pareto efficiency, to obtain a set of valid client/server configurations that balance the maximization of softgoals, rather than maximizing one single softgoal.

[2] Note that leaving these "unknown" in this step would yield the same final result; assigning them now only simplifies the following steps.

A) Obtain a decision vector for requirements distribution. Each possible distribution of N FRs between client and server is stored in a decision vector X_v: $\forall v : 0 \preceq v < 2^N$, $\forall i \in \{1...N\} : X_{v_i} = R_i$, where X_{v_i} is the ith element of X_v, $R_i = S$ if the FR i is allocated on the server and $R_i = C$ if the FR i is allocated on the client.

B) Create a weight matrix. Depending on where the FR is implemented (either on the client or on the server), the contribution of each requirement to each softgoal must be quantified. To this aim, the designer creates a matrix by using the following weights to each kind of contribution: $w = 0$ if the requirement does not contribute to any softgoal, $w = +1$ if there is a "Help" contribution link, $w = -1$ if there is a "Hurt" contribution link, $w = +2$ if there is a "Some +" contribution link, $w = -2$ if there is a "Some -" contribution link, $w = +4$ if there is a "Make" contribution link and $w = -4$ if there is a "Break" contribution link.

Therefore, the matrix is defined, so that each entry $W_{i_j}^k$ corresponds to the contribution of the ith functional requirement to the jth softgoal on the k (client or server) side: $\forall i \in \{1...N\}$, $\forall j \in \{1...M\}$, $\forall k \in \{C, S\}$: $W_{i_j}^k = w$, where N is the number of functional requirements and M is the number of softgoals, and w is defined as previously described.

C) Calculate the objective function. For each softgoal j the corresponding objective function F_j with respect to a decision vector X_v is calculated by summing the contributions of all requirements to each softgoal j taking into account the client/server distribution defined in x_v: $\forall j \in \{1, ..., M\}$, $\forall v \in \{0, ..., 2^N - 1\}$: $F_j(X_v) = \Sigma_{j=1}^{M} W_{i_j}^k$, where N is the number of FRs, M is the number of softgoals.

D) Calculate the overall fitness. The sum of all objective functions with respect to a decision vector X_v is computed to obtain the overall fitness of the decision vector X_v: $\forall j \in \{1...N\}$, $\forall v \in \{0, ..., 2^N - 1\}$: $\Sigma_{j=1}^{M} F_j(X_v)$, where N is the number of requirements and M is the number of softgoals.

E) Calculate the Pareto front. To do this, it is necessary to obtain all decision vectors that are not dominated by any other decision vectors, these forms the Pareto optimal set, while the corresponding objective vectors are said to form the Pareto front.

Step 3- Select the final client/server distribution. From the set of Pareto optimal client/server distributions, the designer now elects the distribution of choice. When making this selection, the designer needs to take into account the priorities of the stakeholder, i.e., which softgoals are most important to realize, and which negative impact on other softgoals is an acceptable consequence. The merit of our approach is that it gives the designer a good overview of the good possible configurations. Furthermore, by using the GORE approach, he also gets a good insight in the impact, both positive and negative, his decisions have on the realization of all softgoals, i.e. on the user satisfaction. Finally, the final RIA requirements model is created using the client/server distribution selected by the designer.

5 Case Study

In this section, we illustrate our approach by presenting an excerpt of a real-world case study of a bioinformatics company. The main goal of this company is to provide online services based on human genome analysis. The company will allow clients to upload a gene, subsequently analyze the gene and provide the user with personalized reports corresponding to the desire of the user. The first and only service that initially will be available is the reporting of potential illnesses the user is vulnerable to, based on the detection of variations of the user's supplied gene compared to the company's gene database. The company is committed to offer their services using a RIA, as they expect a modern Web application will be more attractive for visitors, and they want to study the potential benefit of distributing their functionality between the client and server side. In our excerpt of the case study, we focus (solely) on the reporting requirement to illustrate our approach.

Step 1.- Create the base RIA requirements model. According to our proposal, the first step is to create the initial (base) RIA requirements model. As no Web 1.0 was available, it is necessary to specify the base RIA requirements model from scratch. To do this, the designer has to specify the company's goals and requirements.

In Figure 3 the initial requirements model for this case study is shown, focusing solely on the reporting requirement. The goal of the WebApp (actor) is "Report to be provided". To achieve this goal, some FRs are needed: the navigational requirements "Provide variations by disease" and "Provide disease by variant", the layout requirement "Provide graphical view" and, the service requirements "Provide proprietary format document" and "Provide pdf". Some of these requirements directly or indirectly affect some softgoals, i.e., the service requirement "Provide pdf" affects directly the softgoal "Compatibility" by means of a "Help" contribution and also affects the "Security" softgoal indirectly by means of the goal "Report to be provided".

As explained in Sect. 4, at this point, we do not yet decide which FR will be implemented on the server or on the client side, and thus the contributions to the softgoals are initially labeled as "unknown", as their contribution depends on where they are implemented (e.g. client or server). For the "Compatibility" softgoal however, we see that the contributions are already stated, as those values do not depend on where the requirement is implemented. Contributions originating in "Provide graphical view" are also stated, because this requirement is mandatory implemented on the client, as it can only be realized using client-side technology.

Step 2.- Obtain the Pareto optimal client/server distribution. The aim is to obtain a set of good client/server (FR) distributions. To do this, the following sub-steps are needed.

A) Obtain a decision vector for requirements distribution. To calculate this distributions, we only need to consider those FRs that contribute to the softgoals in the base requirements model, as can be seen in Table 1. Consequently, there are sixteen possible configurations (decision vectors) for the

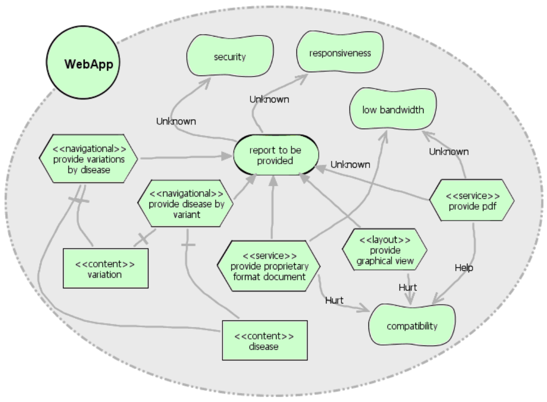


Fig. 3: Base RIA requirements model.

requirements distribution (using 2^4). Table 3 shows all possible decision vectors (column 2 to 5, all rows), in other words, all possible client/server configurations, where C represents Client and S represents Server. The requirement "Provide graphical view" is not considered as it is always allocated client side.

Table 1: *Softgoals* and FRs detected in the base RIA requirements model.

Softgoals	**Requirement**
S1.- security	R1.- provide variations by disease
S2.- responsiveness	R2.- provide disease by variant
S3.- compatibility	R3.- provide pdf
S4.- low bandwidth	R4.- provide proprietary format document

B) Create a weight matrix. Next, we create the weight matrices. For computing the objective functions in this case study, the following matrices are defined: Client matrix (Table 2a) and Server matrix (Table 2b). Each matrix contains the quantification of the contributions made by the FRs to each softgoal, both on the client and on the server side. In practice, the designer will take the base RIA requirements model, and concretize it twice: once filling in the contributions labels under the assumption that all FRs are located on the client, and once assuming they are all located on the server. The first model correspond to the Client matrix (Table 2a), the second to the Server matrix (Table 2b), using the weights defined in Section 4, 2B. As an example, row 3, (requirement "Provide pdf"), column 4 (softgoal "Responsiveness"), shows "+1" in the Client matrix, indicating a "help" contribution if the requirement is located on the client, and shows "-1" in the Server matrix, indicating a "hurt'" contribution if the requirement is located on the server side.

C) Calculate the objective function. The objective functions are calculated using the weight matrices from step 2.B, i.e., for the configuration X1 (Table 3): $F(S1) = -4$ is calculated summing the contributions made by the requirements to the softgoal S1 according to the client weight matrix (Table 2a) and $F(S2) = +4$ is calculated in the same form by using the server weight matrix

(Table 2b). For $F(S1) = -2$ from the configuration X2, the requirements R1, R2 and R3 are implemented in client side, so their contribution is taken from the client weight matrix (Table 2a): $\Sigma(R1, R2, R3) = -3$ and R4 is implemented in the server side, thus, $R4 = +1$ (taken from the server weight matrix (Table 2b), thus $F(S1) = -2$. For this case study, the results of the corresponding objective functions are shown in columns 6 to 9 from Table 3.

D) Calculate the overall fitness. All the objective functions are summed (Σ) with respect to each configuration (decision vector) to obtain the overall fitness (see column 10 from Table 3). For example, in the configuration X1 from Table 3, the overall fitness (Σ) is "+2", i.e. the sum of the softgoals $F(S1) + F(S2) + F(S3) + F(S4)$ for the decision vector X1.

Requirement	S1	S2	S3	S4
R1	-1	+1	0	0
R2	-1	+1	0	0
R3	-1	+1	+1	+1
R4	-1	+1	-1	+1

(a) Client weight matrix.

Requirement	S1	S2	S3	S4
R1	+1	-1	0	0
R2	+1	-1	0	0
R3	+1	-1	+1	-1
R4	+1	-1	-1	-1

(b) Server weight matrix.

Table 2: Weight matrices.

E) Calculate the Pareto front. The configurations (decision vectors) are Pareto front when they are not dominated by any other configurations. It is then said that these configurations form the Pareto optimal set, while the corresponding objective vectors (softgoals) are said to form the Pareto front. For example, the configuration X1 (decision vector X1) is Pareto front, because no other decision vector can be found that satisfies a particular softgoal better, without causing the satisfaction of other softgoals to decrease. Note that $F(S2)$, $F(S3)$ and $F(S4)$ are all maximized in this configuration. On the other hand, the configuration X2 is not Pareto front due to the configuration X5 that better satisfies $F(S4)$, and equally satisfies all other softgoals.

In Table 3, grey rows are the Pareto front from which we can select the final solution according to the priorities we establish over the softgoals. For example, the configuration X1, where every requirement is placed on the client side, is the best option if we do not mind that the satisfaction of softgoal S1 ("Security") is minimal (-4): all other softgoals are maximized, and the overall fitness is +2, indicating that overall, softgoals are well satisfied. On the other hand, if "Security" is the softgoal we want to prioritize, configuration X16, where every requirement is placed on the server side, is best suited. However, it comes with a high price with respect to other softgoals, and has an overall fitness of -2, indicated an overall poor satisfaction of softgoals. All the other solutions of the Pareto front are intermediate configurations that lead us to different tradeoffs such as X6 or X14. Both have an overall fitness of 0, indicating that overall, all softgoals are balanced.

Step 3.- Select the final client/server distribution. In this step, the final RIA requirements model is created using the client/server distribution selected by the designer. For the bioinformatics company, the configuration X14

Table 3: The possible functionality distribution among the server and the client by means of the softgoal tradeoff.

Configuration	R1	R2	R3	R4	F(S1)	F(S2)	F(S3)	F(S4)	Σ	Pareto Front
X1	C	C	C	C	-4	+4	0	+2	+2	Yes
X2	C	C	C	S	-2	+2	0	0	0	No because of X5
X3	C	C	S	C	-2	+2	0	0	0	No because of X5
X4	C	C	S	S	0	0	0	-2	-2	No because of X13
X5	C	S	C	C	-2	+2	0	+2	+2	Yes
X6	C	S	C	S	0	0	0	0	0	Yes
X7	C	S	S	C	0	0	0	0	0	Yes (as same as X6)
X8	C	S	S	S	+2	-2	0	-2	-2	No because of X14
X9	S	C	C	C	-2	+2	0	+2	+2	Yes (as same as X5)
X10	S	C	C	S	0	0	0	0	0	Yes (as same as X6)
X11	S	C	S	C	0	0	0	0	0	Yes (as same as X6)
X12	S	C	S	S	+2	-2	0	-2	-2	No because of X14
X13	S	S	C	C	0	0	0	+2	+2	Yes
X14	S	S	C	S	+2	-2	0	0	0	Yes
X15	S	S	S	C	+2	-2	0	0	0	Yes (as same as X14)
X16	S	S	S	S	+4	-4	0	-2	-2	Yes

(Figure 4) is selected as the final client/server distribution. In X14, the requirements R1, R2 and R4 are placed on the server and R2 on the client (see Table 1). The requirement "Provide graphical view" by default is always allocated in the client side (as mentioned in Step 2.A). This configuration was considered a good tradeoff, because security is deemed important, but other softgoals should not be totally neglected.

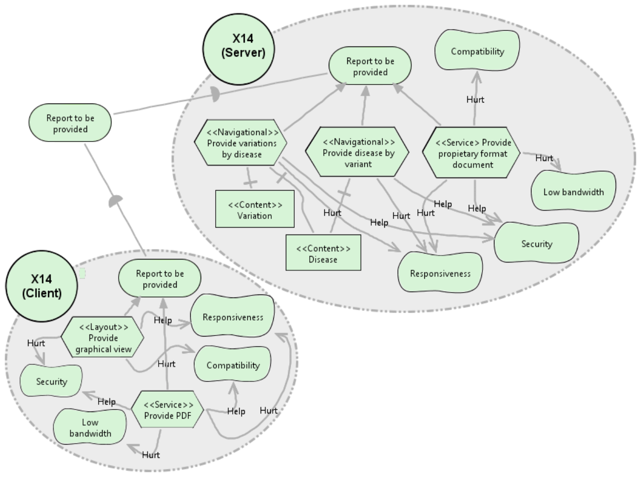
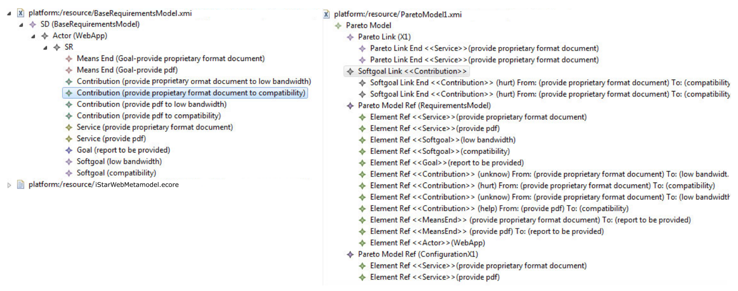


Fig. 4: X14 solution.

6 Implementation

Our approach for client/server distribution of RIA functionality has been implemented by using EMF (http://www.eclipse.org/emf/). EMF is a Java-based modeling framework and code generation facility for building tools and other applications based on a structured data model. In our case, the left-hand side of Fig. 5 shows a representation of the $i*$ requirements model of our case study by using the tree editor of EMF. The right-hand side shows how our Pareto-based approach has been implemented by means of such a tree editor. Furthermore, several Java classes have been developed to obtain a tree representation of the Pareto efficiency to be tailored to the $i*$ model that conforms to our metamodel for specifying Web requirements (previously described in Sect. 3): a *ParetoModel* represents all the solutions of our approach from a specific input requirements model (contained in the *ParetoModelRef*). Both, the *ParetoLink* and *ParetoLinkEnd* reference to each of the elements of this input model in order to be useful for computing the Pareto efficiency.



```
platform:/resource/BaseRequirementsModel.xmi
  SD (BaseRequirementsModel)
    Actor (WebApp)
      SR
        Means End (Goal-provide proprietary format document)
        Means End (Goal-provide pdf)
        Contribution (provide proprietary ormat document to low bandwidth)
        Contribution (provide propietary format document to compatibility)
        Contribution (provide pdf to low bandwidth)
        Contribution (provide pdf to compatibility)
        Service (provide proprietary format document)
        Service (provide pdf)
        Goal (report to be provided)
        Softgoal (low bandwidth)
        Softgoal (compatibility)
platform:/resource/iStarWebMetamodel.ecore
```

```
platform:/resource/ParetoModel1.xmi
  Pareto Model
    Pareto Link (X1)
      Pareto Link End <<Service>>(provide proprietary format document)
      Pareto Link End <<Service>>(provide proprietary format document)
    Softgoal Link <<Contribution>>
      Softgoal Link End <<Contribution>> (hurt) From: (provide proprietary format document) To: (compatibility
      Softgoal Link End <<Contribution>> (hurt) From: (provide proprietary format document) To: (compatibility
    Pareto Model Ref (RequirementsModel)
      Element Ref <<Service>>(provide proprietary format document)
      Element Ref <<Service>>(provide pdf)
      Element Ref <<Softgoal>>(low bandwidth)
      Element Ref <<Softgoal>>(compatibility)
      Element Ref <<Goal>>(report to be provided)
      Element Ref <<Contribution>> (unknow) From: (provide proprietary format document) To: (low bandwidt.
      Element Ref <<Contribution>> (hurt) From: (provide proprietary format document) To: (compatibility)
      Element Ref <<Contribution>> (unknow) From: (provide proprietary format document) To: (low bandwidth
      Element Ref <<Contribution>> (help) From: (provide pdf) To: (compatibility)
      Element Ref <<MeansEnd>> (provide proprietary format document) To: (report to be provided)
      Element Ref <<MeansEnd>> (provide pdf) To: (report to be provided)
      Element Ref <<Actor>>(WebApp)
    Pareto Model Ref (ConfigurationX1)
      Element Ref <<Service>>(provide proprietary format document)
      Element Ref <<Service>>(provide pdf)
```

Fig. 5: The $i*$ base RIA requirements model (left) and the Pareto model (right).

7 Conclusions and Future Work

In this paper, we presented an extension to support RIAs to an existing $i*$-based goal-oriented requirements analysis method for Web 1.0 applications. Our approach goes out from the user's goals and expectations, and allows the designer to evaluate and decide on good client/server distributions for the functionality. To do so, we devised a set of steps based on the Pareto optimal approach that is particularly suited to balance and maximize the conflicting softgoals. Furthermore, it facilitates the evaluation of the obtained (Pareto) optimal solutions and selection of the final solution taking into account the priorities of softgoals. We exemplified our approach using an excerpt of a real-world case study we performed for a bio-informatics company. As future work, we plan to study and incorporate the data distribution between client and server in our Pareto algorithm. We would also like to integrate our approach in a RIA engineering method, and study the possibility to automatically generate design artifacts.

References

1. Aguilar, J.A., Garrigós, I., Mazón, J.N.: Impact Analysis of Goal-Oriented Requirements in Web Engineering. In: ICCSA. pp. 421–436. Springer, Santander, Spain (2011)

2. Aguilar, J.A., Garrigós, I., Mazón, J.N., Trujillo, J.: An MDA Approach for Goal-oriented Requirement Analysis in Web Engineering. Journal of Universal Computer Science (J. UCS) 16(17), 2475–2494 (2010)
3. Aguilar, J.A., Garrigós, I., Mazón, J.N., Trujillo, J.: Web Engineering Approaches for Requirement Analysis - A Systematic Literature Review. In: WEBIST. pp. 187–190 (2010)
4. Bozzon, A., Comai, S., Fraternali, P.: Current Research on the Design of Web 2.0 Applications Based on Model-Driven Approaches. ICWE pp. 25–31 (2008)
5. Collette, Y., Siarry, P.: Multiobjective optimization: principles and case studies. Springer (2003)
6. Cormode, G., Krishnamurthy, B.: Key differences between Web 1.0 and Web 2.0. First Monday 13(6), 2 (2008)
7. Dolog, P., Stage, J.: Designing interaction spaces for Rich Internet Applications with UML. In: Baresi, L., Fraternali, P., Houben, G.J. (eds.) Web Engineering. LNCS, vol. 4607, pp. 358–363. Springer (2007)
8. Horkoff, J., Yu, E.: Evaluating Goal Achievement in Enterprise Modeling–An Interactive Procedure and Experiences. The Practice of Enterprise Modeling pp. 145–160 (2009)
9. Koch, N., Pigerl, M., Zhang, G., Morozova, T.: Patterns for the model-based development of rias. In: ICWE. pp. 283–291. Springer, Berlin (2009)
10. Lowe, D.: Web system requirements: an overview. Requirements Engineering 8, 102–113 (2003)
11. Luna, E.R., Garrigós, I., Mazón, J.N., Trujillo, J., Rossi, G.: An i*-based Approach for Modeling and Testing Web Requirements. Journal of Web Engineering 9(4), 302–326 (2010)
12. Machado, L., Filho, O., Ribeiro, J.a.: UWE-R: an extension to a web engineering methodology for rich internet applications. WSEAS Trans. Info. Sci. and App. 6, 601–610 (April 2009)
13. Meliá, S., Gómez, J., Pérez, S., Díaz, O.: A model-driven development for GWT-based Rich Internet Applications with OOH4RIA. In: ICWE. pp. 13–23. IEEE (2008)
14. Preciado, J.C., Linaje, M., Sanchez, F., Comai, S.: Necessity of methodologies to model rich internet applications. In: WSE. pp. 7–13. IEEE, Washington, DC, USA (2005)
15. Preciado, J., Linaje, M., Comai, S., Sanchez-Figueroa, F.: Designing rich internet applications with web engineering methodologies. In: WSE. pp. 23–30. IEEE (Oct 2007)
16. Preciado, J., Linaje, M., Sanchez-Figueroa, F.: Adapting Web 1.0 User Interfaces to Web 2.0 Multidevice User Interfaces using RUX-Method. Journal of Universal Computer Science (J. UCS) 14(13), 2239–2254 (2008)
17. Sommerville, I.: Software Engineering. Addison-Wesley, 6th edn. (2001)
18. Urbieta, M., Rossi, G., Ginzburg, J., Schwabe, D.: Designing the interface of rich internet applications. In: LA-Web. pp. 144–153. IEEE (2007)
19. Valverde, F., Pastor, O.: Facing the Technological Challenges of Web 2.0: A RIA Model-Driven Engineering Approach. WISE pp. 131–144 (2009)
20. Wright, J.M., Dietrich, J.B.: Requirements for rich internet application design methodologies. In: WISE. pp. 106–119. Springer, Berlin (2008)
21. Wright, J.: A Modelling Language for Interactive Web Applications. In: ASE. pp. 689–692. IEEE (2010)
22. Yu, E.: Modelling Strategic Relationships for Process Reenginering. Ph.D. thesis, University of Toronto, Canada (1995)