A Requirement Analysis Approach for Using i* in Web Engineering

Irene Garrigós, Jose-Norberto Mazón, and Juan Trujillo

Lucentia Research Group
Department of Software and Computing Systems – DLSI
University of Alicante, Spain
{igarrigos,jnmazon,jtrujillo}@dlsi.ua.es

Abstract. Web designers usually ignore how to model real user expectations and goals, mainly due to the large and heterogeneous audience of the Web. This fact leads to websites which are difficult to comprehend by visitors and complex to maintain by designers. In order to ameliorate this scenario, an approach for using the i* modeling framework in Web engineering has been developed in this paper. Furthermore, we also present a traceability approach for obtaining different kind of design artifacts tailored to a specific Web modeling method. Finally, we include a sample of our approach in order to show its applicability and we describe a prototype tool as a proof of concept of our research.

1 Introduction

In the last decade, the number and complexity of websites and the amount of information they offer is rapidly growing. In this context, introduction of Web design methods and methodologies [1,2,3,4,5] have provided mechanisms to develop complex Web applications in a systematic way. To better accommodate the individual user, personalization of websites has been also introduced and studied [6,7,8,9]. However, due to the idiosyncrasy of the audience, traditionally methodologies for Web engineering have not taken into serious consideration the requirement analysis phase. Actually, one of the main characteristics of Web applications is that they typically serve large and heterogeneous audience, since respectively i) everybody can access to the website and ii) each user has different needs, goals and preferences. Interestingly, this is the opposite situation from the traditional software development where the users are well known.

Therefore, current effort for requirement analysis in Web engineering is rather focused on the system and the needs of the users are figured out by the designer. This scenario leads us to websites that do not assure *real* user requirements and goals, thus producing user disorientation and comprehension problems. There may appear development and maintenance problems for designers, since costly, time-consuming and rather non-realistic mechanisms (e.g. surveys among visitors) should be developed to improve the already implemented website, thus increasing the initial project budget.

To solve these drawbacks, in this paper, a complementary viewpoint should be adopted: modeling which are the expectations, intentions and goals of the users when they are browsing the site and determining how they can affect the definition of a suitable Web design. The main benefit of our point of view is that the designer will be able to make decisions from the very beginning of the development phase. These decisions could affect the structure of the envisioned website in order to satisfy needs, goals, interests and preferences of each user or user type. To this aim, we propose to use the i* modeling framework [10,11], one of the most valuable approaches for analyzing stakeholders' goals and how the intended system would meet them. This framework is also very useful for reasoning about how stakeholders' goals contribute to the selection of different design alternatives. However, although i* provides mechanisms to model stakeholders and relationships between them, it should be adapted for Web engineering, since the Web domain has special requirements that are not taken into account in traditional requirement analysis approaches. These requirements are related to the three main features of Web applications [12]: navigational structure, user interface and personalization capability.

Bearing these considerations in mind, in this paper, we present an approach for specifying requirements in the context of a Web engineering method [8] improving the development of Web applications by using i* models. Also we provide the designer with a set of guidelines to define these models. Moreover, the main conceptual models are derived from this requirements specification using QVT (Query/View/Transformation) rules [13]. In this way, designers will not have to create these models from scratch but obtaining a first tentative model that ensures user requirements and then they only have to refine these models, saving time and development effort.

The remainder of this paper is structured as follows: our approach for requirement analysis in Web engineering and how to trace these requirements to the Web design is presented in Sect. 2. Section 3 describes an example of applying our approach. Section 4 describes related work. Finally, in Sect. 5, we present our conclusions and sketch some future work.

2 Modeling Requirements in Web Engineering

In this section, we present a proposal which provides a way of specifying requirements using i* in the context of A-OOH*(Adaptive Object Oriented Hypermedia method)* [8]. A-OOH is the extension of the OO-H modeling method [2], which includes the definition of adaptation strategies. This approach has also been extended with UML-profiles so all the conceptual models are UML-compliant (see Sect. 2.1). We use A-OOH for demonstration purposes but the proposal could be applied to any Web modeling method. Traceability from the specified requirements to the different conceptual models is also studied (see Sect. 2.2). Designers will have to focus on specifying the requirements and will just have to refine the generated conceptual models in order to adjust them.

2.1 Specification of Requirements

The development of Web applications involves different kind of stakeholders with different needs and goals. Interestingly, these stakeholders depend on each other to achieve their goals, perform several tasks or obtain some resource, e.g. *the Web administrator relies on new clients for obtaining data in order to create new accounts.* In the requirements engineering community, goal-oriented techniques, such as the i* framework [10,11], are used in order to explicitly analyze and model these relationships among multiple stakeholders (actors in the i* notation). The i* modeling framework has been proven useful for representing (i) intentions of the stakeholders, i.e. their motivations and goals, (ii) dependencies between stakeholders to achieve their goals, and (ii) the (positive or negative) effects of these goals on each other in order to be able to select alternative designs for the system, thus maximizing goals' fulfilment.

Next, we briefly describe an excerpt of the i* framework which is relevant for the present work. For a further explanation, we refer the reader to [10,11]. 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 interests and concerns and how they might be addressed. The SR model (represented as ◌) provides a detailed way of modeling internal intentional elements and relationships of each actor (○). Intentional elements are goals (⬭), tasks (⬡), resources (▭) and softgoals (☁). Intentional relationships are means-end links (─▷) representing alternative ways for fulfilling goals; task-decomposition links (──┼) representing the necessary elements for a task to be performed; or contribution links ($\xrightarrow[\text{hurt}]{\text{help}}$) in order to model how an intentional element contributes to the satisfaction or fulfillment of a softgoal. A sample application of the i* modeling framework is shown in Fig. 1, which represents the SR model of our case study (see Sect. 3) for the client stakeholder. The main goal of the client is to *"buy books"*. In order to do this, the client should *"choose a book to buy"* and *"provide his/her own data"*. The task *"choose a book to buy"* should be decomposed in several subtasks: *"consult books"*, *"search for a specific book"*, *"consult recommended books"*. These tasks can have positive or negative effects on some important softgoals. For example, while *"consult books"* helps to satisfy the softgoal *"obtain more complete information"*, it hurts the softgoal *"reduce selection time"*. Moreover, *"consult books"* can be further decomposed according to the way in which the book data is consulted.

Although i* provides good mechanisms to model actors and relationships between them, it needs to be adapted to the Web engineering domain to reflect special Web requirements that are not taken into account in traditional requirement analysis approaches, thus being able to assure the traceability to Web design. Web functional requirements are related to three main features of Web applications [12] (besides of the non-functional requirements): navigational structure, user interface and personalization capability. Furthermore, the required data structures of the website should be specified as well as the required (internal)

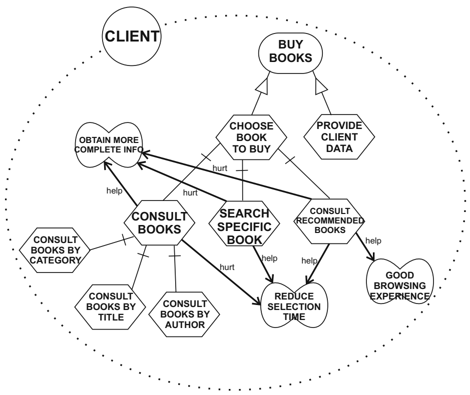


Fig. 1. Modeling the client in an SR model

functionality provided by the system. Therefore, in this paper, we use the taxonomy of Web requirements presented in [12]:

Content Requirements. With this type of requirements the content that the website presents to its users is defined. Some examples might be: “book information” or “product categories”. Other kind of requirements may need to be related with one or more *content requirements.*

Service Requirements. This type of requirement refers to the internal functionality the system should provide to its users. For instance: “register a new client”, “add product”, etc.

Navigational Requirements. A Web system must also define the navigational paths available for the existing users. Some examples are: “consult products by category”, “consult shopping cart”, etc.

Layout Requirements. Requirements can also define the visual interface for the users. For instance: “present a different style for teenagers”, etc.

Personalization Requirements. We also consider personalization requirements in this approach. The designer can specify the desired personalization actions to be performed in the final website (e.g. “show recommendations based on interest”, “adapt font for visual impaired users”, etc.)

Non-Functional Requirements. In our approach the designer can also model non-functional requirements. These kind of requirements are related to quality criteria that the intended Web system should achieve and that can be affected by other requirements. Some examples can be “good user experience”, “attract more users”, “efficiency”, etc.

Once this classification has been adopted, the i* framework needs to be adapted. As the considered Web engineering approach (A-OOH) is UML-compliant, we have used the extension mechanisms of UML to (i) define a profile for using i* within UML; and (ii) extend this profile in order to adapt *i** to specific Web domain terminology. Therefore, new stereotypes have been added

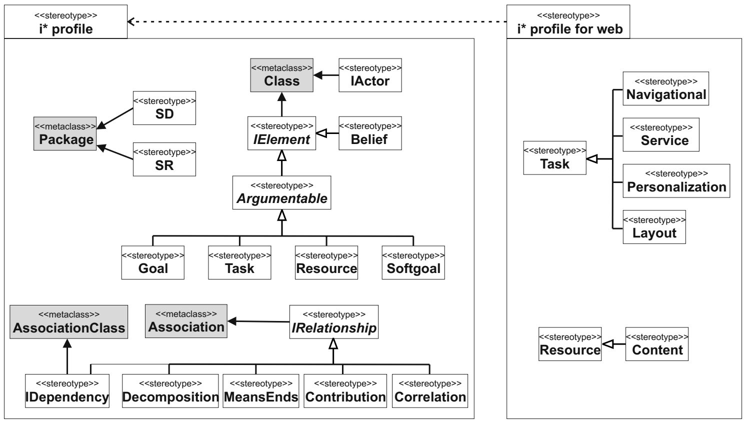


Fig. 2. Overview of the UML profiles for *i** modeling in the Web domain

according to the different kind of Web requirements (see Fig. 2): *Navigational*, *Service*, *Personalization* and *Layout* stereotypes extend the *Task* stereotype and *Content* stereotype extends the *Resource* stereotype. It is worth noting that non-functional requirements can be modeled by directly using the softgoal stereotype.

Finally, several guidelines should be provided in order to support the designer in defining i* models for Web domain.

1. Determine the kind of users for the intended Web and model them as actors. The website is also considered as an actor. Dependencies among these actors must be modeled in an SD model.
2. Define actors' intentions by using i* techniques in an SR model [14]: modeling goals, softgoals, tasks and resources, and the relationships between them.
3. Annotate tasks as navigational, service, personalization or layout requirements. Also, annotate resources as content requirements. It is worth noting that goals and softgoals should not be annotated.

2.2 Traceability to Web Design

Once the requirements have been defined they can be used to derive the conceptual models for the website. Typically, Web design methods have three main models to define a Web application: a *Domain model*, in which the structure of the domain data is defined, a *Navigation model*, in which the structure and behavior of the navigation view over the domain data is defined, and finally a *Presentation model*, in which the layout of the generated hypermedia presentation is defined. To be able to model personalization at design time two additional models are needed: a *Personalization model*, in which personalization strategies are specified, and a *User model*, in which the structure of information needed for personalization is described.

As aforementioned, the conceptual models of the A-OOH method are derived from requirements. Once these models are derived the designer has only to refine them, avoiding the task of having to create them from scratch. Due to space constraints, in this work, the focus is on the Domain and Navigation models. However, an skeleton of the Presentation, User and Personalization models could also be generated from the requirements specification.

Since the i* framework does not well support traceability to other design artifacts by its own, domain-oriented mechanisms should be considered to perform this task [15]. In our approach, the new stereotypes presented in the previous subsection allow us to prepare models for this traceability phase. We have detected several i* patterns [16] in order to define a set of QVT transformation rules to map elements from the SR metamodel to their counterparts in the A-OOH metamodel. They are applied with a certain order as shown in Fig. 3, where the transformation workflow is summarized.

After analyzing and modeling the requirements of the website according to the guidelines presented in the previous subsection, the Domain model (DM) and Navigational model (NM) are generated from the specified requirements. Before explaining each of the derivations, we briefly introduce the A-OOH DM and NM so the reader can easily follow the derivation of them.

Deriving the Domain model. The A-OOH DM is expressed as a UML-compliant class diagram. It encapsules the structure and functionality required of the relevant concepts of the application and reflects the static part of the system. The main modeling elements of a class diagram are the classes (with their attributes and operations) and their relationships.

Table 1 summarizes how DM elements are mapped from the SR model. To derive a preliminary version of the DM we take into account two types of requirements defined in Sect. 2 content and service requirements. We have detected several patterns in the i* models and we have used these patterns to define several

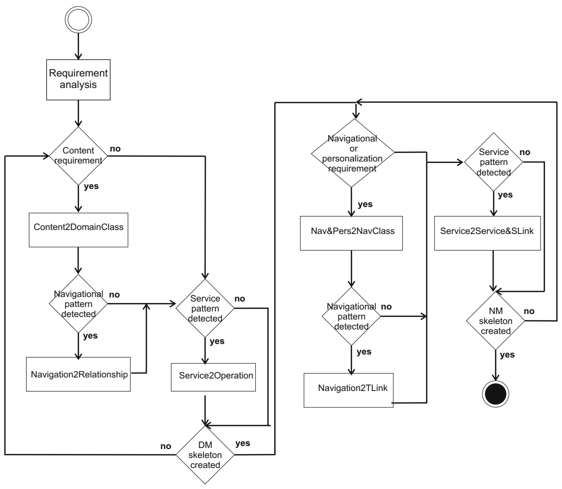
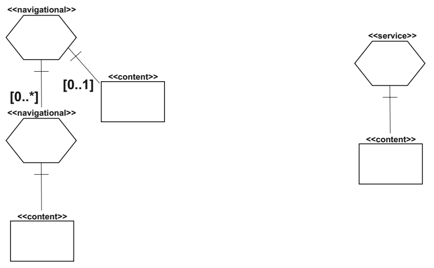


Fig. 3. Transformation Rules: Traceability to Web design

Table 1. Derivation of the Domain model

i* element	**A-OOH element**
Content Requirement	Class
Service Pattern	Operation
Navigational Pattern	Association between classes



(a) Navigational pattern (b) Service pattern

Fig. 4. Patterns

transformation rules in QVT. Specifically, three transformation rules are defined in order to derive the DM from the SR model:

- *Content2DomainClass* By using this transformation rule, each content requirement is detected and derived into one class of the DM.
- *Navigation2Relationship* Preliminar relations into classes are derived from the relations among goals/tasks with attached resources by applying this rule. To generate the associations in the DM we have to detect a *navigational pattern* in the SR model of the *website* stakeholder. In Fig. 4(a) we can see that the *navigational pattern* consists of a navigational root requirement (i.e. task) which can contain one or more navigational requirements attached. Each of the navigational requirement can have attached a resource (i.e. content requirement). The classes mapped from the resources we find in such pattern will have an association relation between them.
- *Service2Operation* This transformation rule detects a *service pattern*, i.e. a service requirement with an attached content requirement in the SR model (see Fig. 4(b)). In this case each service requirement is transformed into one operation of the corresponding class (represented by the content requirement). In this QVT rule (shown in Fig. 5), a service pattern is detected and transformed into the corresponding elements in the target model.

Once the DM skeleton has been obtained it is left to the designer to refine it, who will also have to specify the most relevant attributes of the classes, identify the cardinalities and define (if existing) the hierarchical relationships.

After the preliminar DM is created, a skeleton of the NM is also derived from the specified requirements. This diagram enriches the DM with navigation and interaction features. It is introduced next.

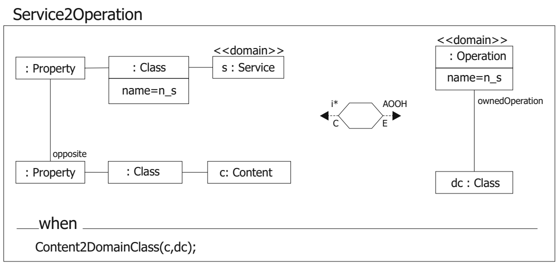


Fig. 5. QVT transformation rule for the service pattern

Deriving the Navigational model. The A-OOH Navigational model is composed of Navigational Nodes, and their relationships indicating the navigation paths the user can follow in the final website (Navigational Links).

There are three types of Nodes: (a) Navigational Classes (which are view of the domain classes), (b) Navigational Targets (which group the model elements which collaborate in the fulfilment of every navigation requirement of the user) and (c) Collections (which are (possible) hierarchical structures defined in Navigational Classes or Navigational Targets. The most common collection type is the C-collection (Classifier collection) that acts as an abstraction mechanism for the concept of menu grouping Navigational Links). Navigational Links (NL) define the navigational paths that the user can follow through the system. A-OOH defines two main types of links: Transversal links (which are defined between two navigational nodes) and Service Links(in this case navigation is performed to activate an operation which modifies the business logic and moreover implies the navigation to a node showing information when the execution of the service is finished).

To derive the NM we take into account the content requirements, service requirements and the navigation and personalization requirements. We also take into consideration the patterns detected (see Fig. 4) in order to develop several QVT transformation rules. In Tab. 2 we can see a summary showing how the different requirements are derived into elements of the NM. In the right part of Fig. 3 we can see the different transformation rules that are to be performed in order to derive a preliminar Navigation model. In this case we also define three transformation rules:

– *Nav&Pers2NavClass:* By using this rule, a “home” navigational class is added to the model, which is a C-collection representing a Menu grouping navigational links. From each navigational and personalization requirement with an associated content requirement a navigational class (NC) is derived. From the “home” NC a transversal link is added to each of the generated NCs.
– *Navigation2TLink:* This rule checks the *navigational pattern*, if it is detected, then a transversal link is added from the NC that represents the root navigational requirement to each of the NCs representing the associated navigational requirements.

Table 2. Derivation of the Navigation model

i* element	A-OOH element
Navigation and Personalization Requirements	Navigational Class
Navigation Pattern	Transversal Links
Service pattern	Operation + Service Link with a target Navigational Class

– *Service2Service&SLink:* Finally, the *service pattern* is checked by applying this transformation rule. If a service pattern is found, then an operation to the class representing the resource is added and service link is created from each of the operations, with a target navigational class which shows the termination of the service execution.

Finally, the derived NM could be refined by the designer in order to specify complementary elements for the desired navigation paths.

3 Sample Application of Our Approach

In this section, we provide an example of our approach based on a company that sells books on-line. In this case study, a company would like to manage book sales via an online bookstore, thus attracting as many clients as possible. Also there is an administrator of the Web to manage clients.

3.1 Requirements Specification

Three actors are detected that depend on each other, namely *"Client"*, *"Administrator"*, and *"Online Bookstore"*. A client depends on the online bookstore in order to *"choose a book to buy"*. The administrator needs to use the online bookstore to *"manage clients"*, while the *"client data"* is provided by the client. These dependencies are modeled by an SD model (see Fig. 6). Once the actors have been modeled in an SD model, their intentions are specified in SR models.

The SR model for the client actor was previously explained in Sect. 2.1. The SR model of the online bookstore is shown in Fig. 6. The main goal of this actor is to *"manage book sales"*. To fulfill this goal the SR model specifies that two tasks should be performed: *"books should be sold online"* and *"clients should be managed"*. We can see in the SR model that the first of the tasks affects positively the softgoal *"attract more users"*. Moreover, to complete this task four subtasks should be obtained: *"provide book info"* (which is a navigational requirement), *"provide recommended books"* (which is a personalization requirement), *"search engine for books"*, and *"provide a shopping cart"*. We can observe that some of these tasks affect positively or negatively to the non-functional requirement *"easy to maintain": "Provide book information"* is easy to maintain, unlike *"provide recommended books"* and *"use a search engine for books"*. The navigational requirement *"provide book information"* can be decomposed into several navigational requirements according to the criteria used to sort the data. These data

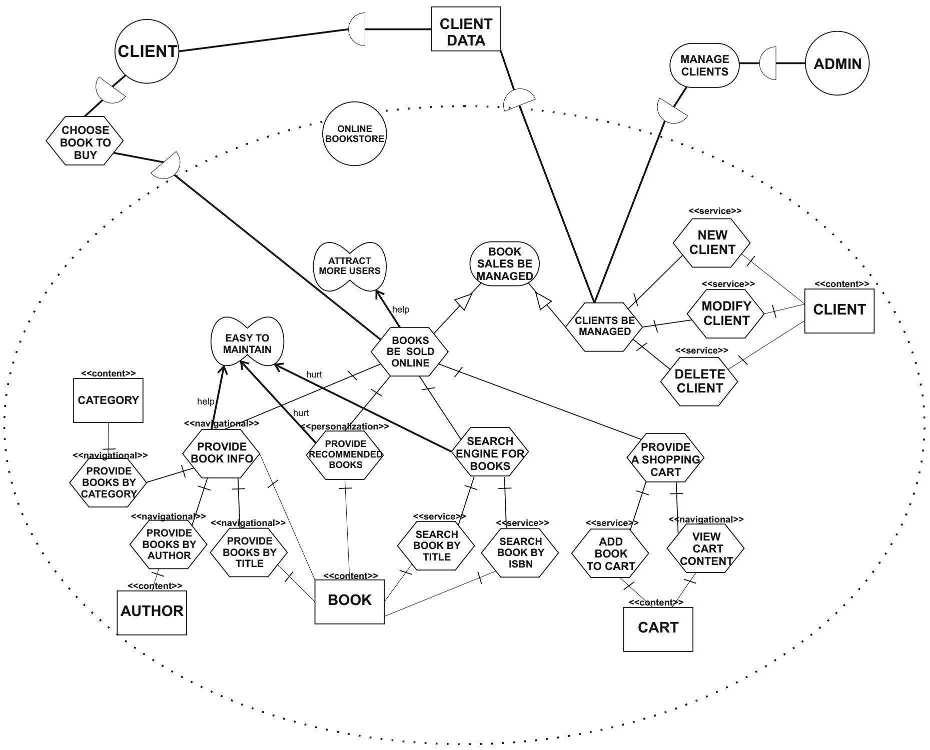

CLIENT
CLIENT DATA
MANAGE CLIENTS
ADMIN
CHOOSE BOOK TO BUY
ONLINE BOOKSTORE
ATTRACT MORE USERS
BOOK SALES BE MANAGED
<<service>> NEW CLIENT
<<service>> MODIFY CLIENT
<<content>> CLIENT
CLIENTS BE MANAGED
<<service>> DELETE CLIENT
help
EASY TO MAINTAIN
BOOKS BE SOLD ONLINE
hurt
<<content>> CATEGORY
help
hurt
<<navigational>> PROVIDE BOOK INFO
<<personalization>> PROVIDE RECOMMENDED BOOKS
SEARCH ENGINE FOR BOOKS
PROVIDE A SHOPPING CART
<<navigational>> PROVIDE BOOKS BY CATEGORY
<<navigational>> PROVIDE BOOKS BY AUTHOR
<<navigational>> PROVIDE BOOKS BY TITLE
<<service>> SEARCH BOOK BY TITLE
<<service>> SEARCH BOOK BY ISBN
<<service>> ADD BOOK TO CART
<<navigational>> VIEW CART CONTENT
<<content>> AUTHOR
<<content>> BOOK
<<content>> CART

Fig. 6. Modeling the online bookstore in an SR model and the SD model

is specified by means of content requirements: *"book", "author"* and *"category"*. The personalization requirement *"provide recommended books"* is related to the content requirement *"book"* because it needs the book information to be fulfilled. The task *"search engine for books"* is decomposed into a couple of service requirements: *"search book by title"* and *"search book by ISBN"*, which are also related to the content requirement *"book"*. In the same way, the task *"provide a shopping cart"* is decomposed into two service requirements: *"add book to cart"* and *"view cart content"*. These service requirements are related to the content requirement *"cart"*. Finally, the task *"clients be managed"* is decomposed into three service requirements: *"new client", "modify client"* and *"delete client"*, which are related to the content requirement *"client"*.

3.2 Traceability to Domain and Navigational Models

In Fig. 7 we can see the derived Domain model from the specified requirements. As explained in Sect. 2.2 to derive the Domain model we take into account the content and service requirements as well as the existence of service or navigational patterns. In this case we can see that five domain classes are created by applying the *Content2DomainClass* transformation rule: one class is generated for each content requirement specified in the SR model. Moreover, we detect three service patterns (see Fig. 4(b)), so operations are added to the classes *client, cart* and *book* by executing the *Service2Operation* rule. Finally we detect that the *Provide Book Info* requirement follows the navigational pattern as we can see in Fig. 4(a). In this case the rule *Navigation2Relationship* adds associations among all the resources found in this pattern. The generated Domain model is shown in Fig. 7.

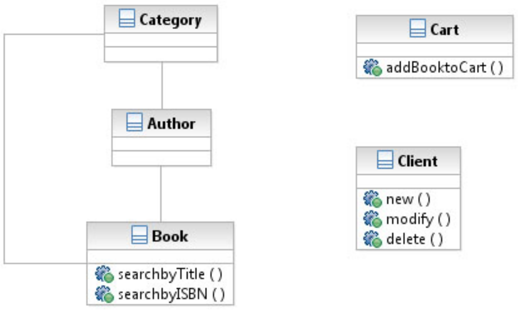


Fig. 7. Traceability to Domain model

In the case of the Navigational model, the rule *Nav&Pers2NavClass* is performed adding a home page with a collection of links (i.e. menu). Afterwards, one NC is created for each navigational and personalization requirement with an attached resource, in this case we have five NC created from navigational and

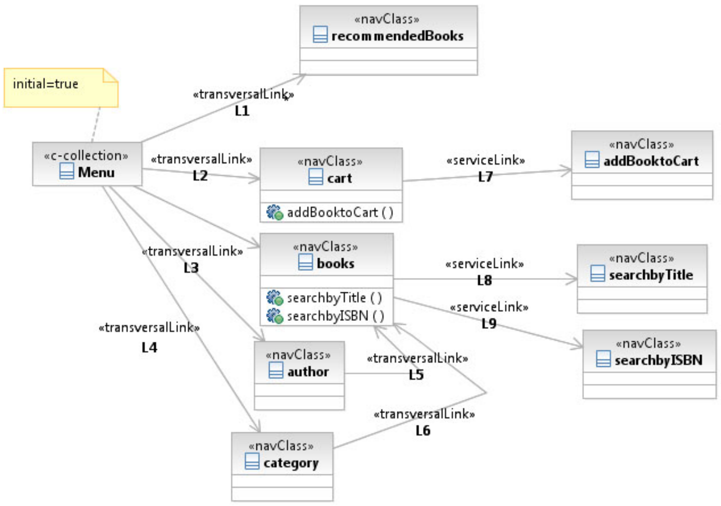


Fig. 8. Traceability to Navigation model

personalization requirements. From the menu, a transversal link to each of the created NCs is added (L1 to L4).

The next step is checking the navigational and service patterns. In this example, we find a navigational pattern (see Fig.4(a)) applying the *Navigation2TLink* it implies creating a transversal link from the NCs created by the associated navigational requirements, to the NC that is represented by the root navigational requirement. In this case two links are added: L5 and L6.

Finally, as we are referring to the website stakeholder, we find three service patterns from which the operations of the NCs books and cart are added and the service links L7, L8 and L9 are created with an associated target NC by applying the *Service2Service&SLink*.

3.3 Implementation Framework

The presented approach has been implemented by using the *Eclipse development platform* [17]. *Eclipse* is a framework which can be extended by means of plugins in order to add more features and new functionalities. A plugin that supports both of the defined profiles has been developed. This new plugin implements several graphical and textual editors (Fig. 9 shows an overview of the tool). The palette for drawing the different elements of i* can be seen on the right-hand side of this figure. A sample SR model is shown in the center of the figure. Traceability rules are also being defined and tested in our prototype.

4 Related Work

Few approaches have focused on defining an explicit requirement analysis stage to model the user. We can stress the following:

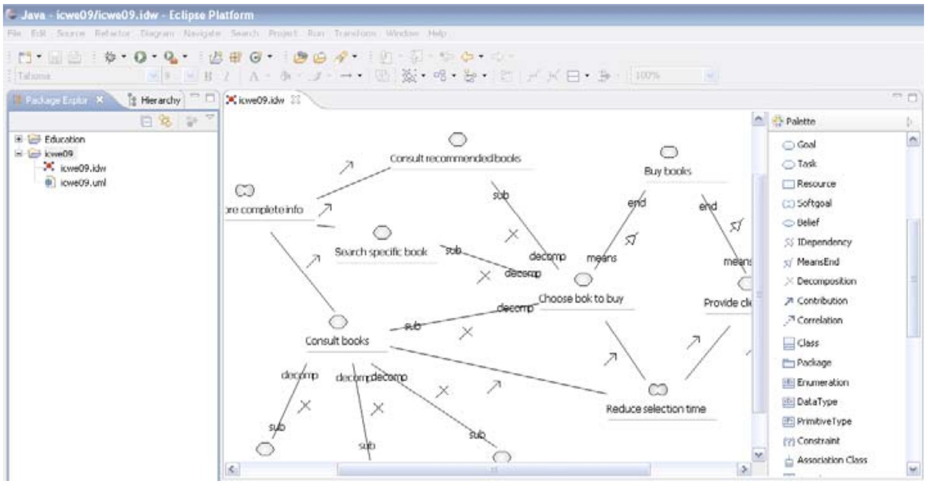


Fig. 9. Screenshot of our prototype

NDT[12] considers a complete taxonomy for the specification of Web requirements. It allows to specify requirements by means of use cases diagrams and templates. It uses a different template for each requirement type they consider, so requirements and objectives are described in a structured way. UWE [7] also describes a taxonomy for requirements related to the Web. It proposes extended use cases, scenarios and glosaries for specifying requirements. WebML [5] also proposes the use of use case diagrams combined with activity diagrams and semi-structured textual description. WSDM [3] is an audience driven approach in which they do a classification of the requirements and the audience. These classes are represented with a diagram in which they are related. Then they are modeled into detail in a Task model using concurrent task trees. OOHDM [18] capture the requirements in use case diagrams. They propose the use of UIDs (user interaction diagrams) for defining the requirements deriving them from the Use cases. OOWS [19] focuses on the specification of tasks. They extend the activity diagrams with the concept of interaction point to describe the interaction of the user with the system.

Furthermore traceability from the requirements to conceptual models is an important issue to bridge the gap between requirements and Web design. There are two approaches to the author's knowledge that support this in some way: OOWS provides automatic generation of (only) navigation models from the tasks description by means of graph transformation rules. NDT [20] defines a requirement metamodel and allows to transform the requirements model into a content and a navigational model by means of QVT rules. Our approach of traceability resembles NDT since we have also adopted QVT in order to obtain design artifacts from Web requirements, but we have kept the benefits of the i* framework by means of the defined profiles and patterns.

However, some of these approaches present the following drawbacks: (i) they do not take into consideration a complete taxonomy of requirements which is suitable in Web applications, or (ii) they consider non-functional requirements in an isolated manner, or (iii) they mainly focus on design aspects of the intended Web system without paying enough attention to Web requirements. Furthermore, none of them perform the analysis of the users' needs. Requirements are figured out by the designer, it may be needed to re-design the website after doing usability and satisfaction tests to the users. Modeling users allow us ensuring that the Web application satisfies real user needs and goals and the user is not overwhelmed with functionalities that he does not need or expect and he does not miss functionalities that were not implemented.

To the best of our knowledge, the only approaches that use goal oriented techniques have been presented in [21,22]. They propose a complete taxonomy of requirements for the Web and use the i* notation to represent them. Unfortunately, they do not benefit from every i* feature, since they only use a metamodel that has some of its concepts, e.g. means-end, decomposition or contribution links from i* are not specified in the approach presented in [21].

5 Conclusions and Future Work

Websites require special techniques for requirement analysis in order to reflect, from early stages of the development, specific needs, goals, interests and preferences of each user or user type. However, Web engineering field does not pay the attention needed to this issue. We have presented a goal oriented approach on the basis of the i* framework to specify Web requirements. It allows the designer to make decisions from the very beginning of the development phase that would affect the structure of the envision website in order to satisfy users.

Moreover, the following guidelines are provided to the designer to properly define i* models for the Web domain: (i) discovering the intentional actors (i.e. Web users and the Web application) and their dependencies in an SD model, (ii) discovering their intentional elements, thus defining SR models for each one, and (iii) annotating intentional elements with Web concepts. We can use this model to check the current website or to make the appropriate decision to build a new one. Moreover, we have defined a set of transformation rules in order to assure the traceability from requirements to the conceptual models. Although this approach is presented in the context of the A-OOH modeling method it can be applied to any Web modeling approach.

Our short-term future work consists of completing the transformation rules in order to obtain the rest of the A-OOH models (i.e. presentation and personalization models).

Acknowledgements. This work has been partially supported by the ESPIA project (TIN2007-67078) from the Spanish Ministry of Education and Science, and by the QUASIMODO project (PAC08-0157-0668) from the Castilla-La Mancha Ministry of Education and Science (Spain). Jose-Norberto Mazón is funded by the Spanish Ministry of Education and Science under a FPU grant (AP2005-1360).

References

1. Casteleyn, S., Woensel, W.V., Houben, G.J.: A semantics-based aspect-oriented approach to adaptation in web engineering. In: Hypertext, pp. 189–198 (2007)
2. Cachero, C., Gómez, J.: Advanced conceptual modeling of web applications: Embedding operation interfaces in navigation design. In: JISBD, pp. 235–248 (2002)
3. Casteleyn, S., Garrigós, I., Troyer, O.D.: Automatic runtime validation and correction of the navigational design of web sites. In: Zhang, Y., Tanaka, K., Yu, J.X., Wang, S., Li, M. (eds.) APWeb 2005. LNCS, vol. 3399, pp. 453–463. Springer, Heidelberg (2005)
4. Koch, N.: Software engineering for adaptive hypermedia systems: Reference model, modeling techniques and development process. Softwaretechnik- Trends 21(1) (2001)
5. Ceri, S., Manolescu, I.: Constructing and integrating data-centric web applications: Methods, tools, and techniques. In: VLDB, p. 1151 (2003)
6. Rossi, G., Schwabe, D., Guimarães, R.: Designing personalized web applications. In: WWW, pp. 275–284 (2001)

7. Koch, N.: Reference model, modeling techniques and development process software engineering for adaptive hypermedia systems. KI 16(3), 40–41 (2002)
8. Garrigós, I.: A-OOH: Extending Web Application Design with Dynamic Personalization. Ph.D thesis, University of Alicante, Spain (2008)
9. Daniel, F., Matera, M., Morandi, A., Mortari, M., Pozzi, G.: Active rules for runtime adaptivity management. In: AEWSE (2007)
10. Yu, E.: Modelling Strategic Relationships for Process Reenginering. Ph.D thesis, University of Toronto, Canada (1995)
11. Yu, E.: Towards modeling and reasoning support for early-phase requirements engineering. In: RE, pp. 226–235 (1997)
12. Cuaresma, M.J.E., Koch, N.: Requirements engineering for web applications - a comparative study. J. Web Eng. 2(3), 193–212 (2004)
13. QVT Language: `http://www.omg.org/cgi-bin/doc?ptc/2005-11-01`
14. i* wiki: `http://istar.rwth-aachen.de`
15. Estrada, H., Rebollar, A.M., Pastor, O., Mylopoulos, J.: An empirical evaluation of the * framework in a model-based software generation environment. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 513–527. Springer, Heidelberg (2006)
16. Strohmaier, M., Horkoff, J., Yu, E.S.K., Aranda, J., Easterbrook, S.M.: Can patterns improve i* modeling? two exploratory studies. In: Paech, B., Rolland, C. (eds.) REFSQ 2008. LNCS, vol. 5025, pp. 153–167. Springer, Heidelberg (2008)
17. Eclipse: `http://www.eclipse.org/`
18. Schwabe, D., Rossi, G.: An object oriented approach to web-based applications design. TAPOS 4(4), 207–225 (1998)
19. Valderas, P., Pelechano, V., Pastor, O.: A transformational approach to produce web application prototypes from a web requirements model. Int. J. Web Eng. Technol. 3(1), 4–42 (2007)
20. Koch, N., Zhang, G., Cuaresma, M.J.E.: Model transformations from requirements to web system design. In: ICWE, pp. 281–288 (2006)
21. Bolchini, D., Paolini, P.: Goal-driven requirements analysis for hypermedia-intensive web applications. Requir. Eng. 9(2), 85–103 (2004)
22. Molina, F.M., Pardillo, J., Álvarez, J.A.T.: Modelling web-based systems requirements using wrm. In: WISE Workshops, pp. 122–131 (2008)