CONFERENCE REVIEW SYSTEM: A CASE OF STUDY

Cristina Cachero[1], Jaime Gómez[1], Antonio Párraga[1] and Oscar Pastor[2]
[1] **Departamento de Lenguajes y Sistemas Informáticos**
Universidad de Alicante. SPAIN
[ccachero, jgomez, aparraga]@dlsi.ua.es
[2] **Departamento de Sistemas Informáticos y Computación**
Universidad Politécnica de Valencia. SPAIN
opastor@dsic.upv.es

Abstract This report presents the OO-H solution to the Conference Review System case study proposed in the IWWOST'01. We first introduce the main concepts of OO-H. We then perform an analysis of the case study, and establish the functional requirements each actor's interface should fulfil. We also present the UML use-case and class diagrams that constitute the basis on which OO-H defines its own models. Next, following the OO-H notation, we present the interface navigation model for each actor of the system. We also briefly illustrate how these models are transformed into an XML specification that feeds a model compiler, capable of generating an operative version of the modelled interface.

1.- A brief introduction to OO-H

The OO-H (Object Oriented Hypermedia) Method is a generic model, based on the object oriented paradigm, that provides the designer with the semantics and notation necessary for the development of web-based interfaces and its connection with previously existing application logic modules.

OO-H defines a set of diagrams, techniques and tools that shape a sound approach to the modelling of web interfaces. The OO-H proposal includes:

- a Design Process
- a Pattern Catalog
- a Navigation Access Diagram (NAD)
- an Abstract Presentation Diagram (APD)
- a CASE tool that allows to automate the development of web applications

In this paper we will focus on the views OO-H defines to extend those provided by 'traditional software' production environments, namely (1) the Navigational Access Diagram (NAD), that defines a navigation view, and (2) the Abstract Presentation Diagram (APD), that gathers the concepts related to abstract presentation. The NAD diagram enriches the domain view provided in the UML [5] use-cases and class diagrams with navigation and interaction features. Also, to define navigation and visualization constraints, OO-H uses the Object Constraint Language (OCL [6]), a subset of the standard UML that allows software developers to write constraints over object models. OO-H associates such constraints to the NAD models by means of filters, that will be explained below. On the other hand, the definition of abstract pages in the APD is based on a set of XML DTD's [1]. Both the NAD and the APD capture

the interface-related design information with the aid of a set of patterns, defined in an Interface Pattern Catalog that is integrated in the OO-H proposal.

The navigation model is captured by means of one or more NAD's. The designer should construct as many NAD's as different views of the system are required, and she should provide at least a different NAD for each user-type (agent-type) who is allowed to navigate through the system. This diagram is based on four types of constructs: (1) Navigation Classes, (2) Navigation Targets, (3) Navigation Links and (4) Collections. Also, when defining the navigation structure, the designer must take into account some orthogonal aspects such as the desired navigation behaviour, the object population selection, the order in which objects should be navigated or the cardinality of the access. These features are captured by means of different kinds of navigation patterns and filters associated with links and collections. In Table 1 we present an overview of the main NAD constructs.

- **Navigation Classes** (NC): they are enriched domain classes whose attribute and method visibility has been restricted according to the user access permissions and navigation requirements. A sample enrichment is the differentiation among three types of attribute: V-Attributes (Visible Attributes), R-Attributes (Referenced Attributes, which are displayed after a user demand) and H-Attributes (Hidden Attributes, only displayed when an exhaustive system population view is required, e.g. for code refinement reasons).

Navigation Targets				
Grouping mechanism to provide cohesive views of elements collaborating in the coverage of related user requirements.				
Collections				
(Possibly) hierarchical structures defined on navigation classes or navigation targets that provide extra paths to information.				
Classes				
Enriched domain classes whose attributes and method visibility are restricted according to the user access permissions and navigation requirements.				
Links				
Navigation construct that provides a controlled access to the target information/navigation structures (classes, collections, methods and so on). Links define both the paths the user may follow through the system and the way they are going to traverse suc				
TYPE	Requirement || Internal || Traversal || Service || Response || Exit			
VISUALIZATION	Origin || Destination			
USER INTERACTION	Manual || Automatic			
APPLICATION SCOPE	Simple || Multiple || Universal			
ACTIVATION LINKS	Links at level n-1 in the navigation path that activate the actual link (level n), that is, make it available to the user.			
Filters			**Patterns**	
Origin	Destination		Index (yes/no)	Navigation (yes/no)
	User-defined	Domain-dependent	Items per page	Items per page

To userType.navigationTarget.construct

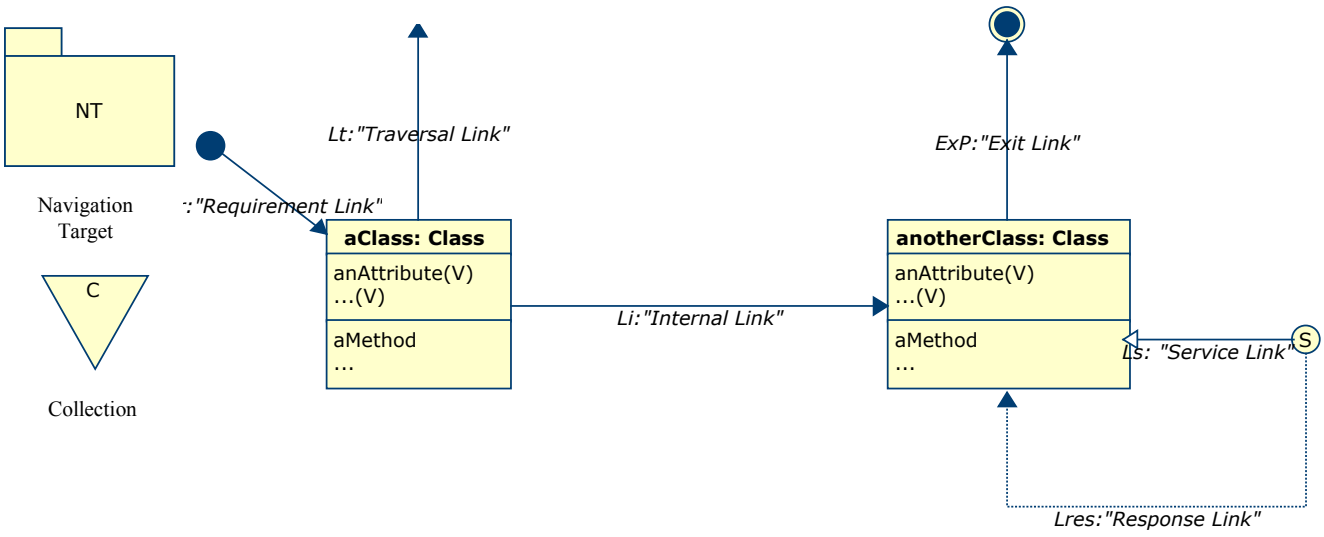

Table 1: OO-H Constructs, icons and navigation-related metamodel attributes

- **Navigation Targets** (NT): they group the elements of the model that collaborate in the coverage of each user navigation requirement.

- **Navigation Links** (NL): they define the navigation paths the user is able to follow through the system. They may have both a Navigation Pattern and/or a set of Navigation Filters associated, which together provide the required additional information to construct the user navigation model. OO-H defines six link types:

    - **I-Links** (Internal Links) define the navigation path inside the boundaries of a given NT.
    - **T-Links** (Traversal Links) are defined between navigation classes belonging to different NT.
    - **R-Links** (Requirement Links) point at the starting navigation point inside each NT.
    - **X-Links** (Exit Links) point at places outside the boundaries of the application. They are also used as an auxiliary mechanism to represent the feeding of parameters to methods.
    - **S-Links** (Service Links) and their corresponding **R-Links** (Response Links) show the services available to the user type associated to that NAD and the view the user accesses when the interface recovers the control of the application. Service links also gather the way the user is required to introduce the parameters needed for the invocation of any method. Regarding such parameter introduction, OO-H defines five possibilities: (1)'hidden' and (2)'constant' parameters imply no user introduction of values. By default the introduction mode is set to (3)'immediate', which means that the interface shows a text field where the user must type the required value. When the user is allowed to choose among a predefined set of possibilities the introduction mode is set to (4) 'selection'. Last, when the parameter selection requires navigation, the (5)'navigation' mode (with a start navigation link and an end navigation link, chosen among those defined in any NAD) is established for that parameter.

- **Collections**: they are (possibly) hierarchical structures defined on Navigation Classes or Navigation Targets. They provide the user with new ways of accessing the information. The most common type of collection, and the one we will use along this paper, is the C-Collection (Classifier Collection), that acts as an abstraction mechanism for the 'menu' concept.

Regarding Navigation Filters, we have already mentioned that they are captured in OO-H by means of OCL expressions. We can distinguish between filters applied to objects in origin (Fo) and filters applied to target population (Fd). Fo's are useful to capture navigation constraints that imply the user is only allowed to continue navigating if the origin population fulfils certain conditions. In our example (see Fig. 8) going to the '*New Conference*' view is only permitted if there is no conference defined (*Fo: Conference.population==0*). On the other hand, Fd's are useful to restrict the views the user has on the target object population. As an example, the navigation requirement 'consult the data about the papers that a given author has introduced' requires the definition of a Fd based on the structural relationship between author and paper. Note

however that Fd's do not necessary have to be defined after structural relationships. For example, 'view all authors affiliated to the Univesity of Alicante', supposes the definition of an Fd based on the *Author.affiliation* attribute. As the reader can infer, the main difference between Fo's and Fd's is that, while Fo's inhibit navigation (the appearance of links in the interface), Fd's restrict the target population being visualized, but don't refrain the link itself from being shown in the interface.

On the other hand, Navigation Patterns are characterized by two properties: indexing (yes/no and, if yes, number of items per page, in order to allow index pagination) and navigation (yes/no and, if yes, number of items per page, in order to diminish guided tours size).

OO-H defines other navigation-related metamodel attributes associated to links, namely:

- Visualization (show in origin | show in destination): any link implicitly connects an origin (either implicit or explicit) and a target information set. When the visualization attribute is set to origin, the target information set is visualized together with the origin, that is, in the same abstract page. However, when it is set to destination, a new abstract page is generated and a navigation action (such as clicking on an anchor) is required to follow the navigation path.
- User Interaction (manual | automatic): Sometimes it is useful for the user not to be obliged to click on a link in order to get the target information set. This characteristic is captured in the 'User Interaction' metamodel attribute, which in this case will be set to 'automatic'.
- Application Scope (simple | multiple | universal): This concept stands for the number of objects a given links involves in origin when it is traversed. The origin of a given link can be defined to be single object (simple), a set of objects chosen by the user (multiple) or the set of objects present in the actual view (universal).

Last but not least, OO-H introduces the concept of activation link. Several times the information the user needs to access slightly varies depending on the contextual navigation (where s/he comes from). OO-H abstracts such situation by means of an activation-link mechanism. Each link defines its set of activation links, that is, links that, when coming through them, make the actual link available to further navigation. All this concepts will be used in section 4, when we present the Navigation Access Diagrams (NAD from now on) corresponding to the Conference Review System example. Commercial interfaces tend to require a greater level of sophistication than that provided by the NAD diagram, regarding both appearance and usability features. In order to refine the interface, OO-H defines another diagram: the Abstract Presentation Diagram (APD), that will be briefly introduced in section 5, once the navigation diagrams for the case study have been presented. Furthermore, and although it is not defined inside the OO-H method, the OO-H CASE tool includes a third view, the CLD (Composite Layout Diagram) that allows the visual manipulation of the final XML interface specification (new frames, styles, and so on) that feeds the model compiler in order for it to generate an operative interface.

More detailed information both on the semantics of the different constructs and on the OO-H process can be found in [3,4].

2.- Use Case Diagrams

The Use Case Diagram is one of the key mechanisms of UML. It captures the system functional requirements for each user type (actor), and drives the remaining phases of the software construction process. OO-H uses it as a basis on which the navigation requirements are structured.

Departing from the paper review system description, we have defined four use case diagrams, one for each actor identified in the system, namely PCChair, PCMember, Reviewer and Author.

2.1. PCChair

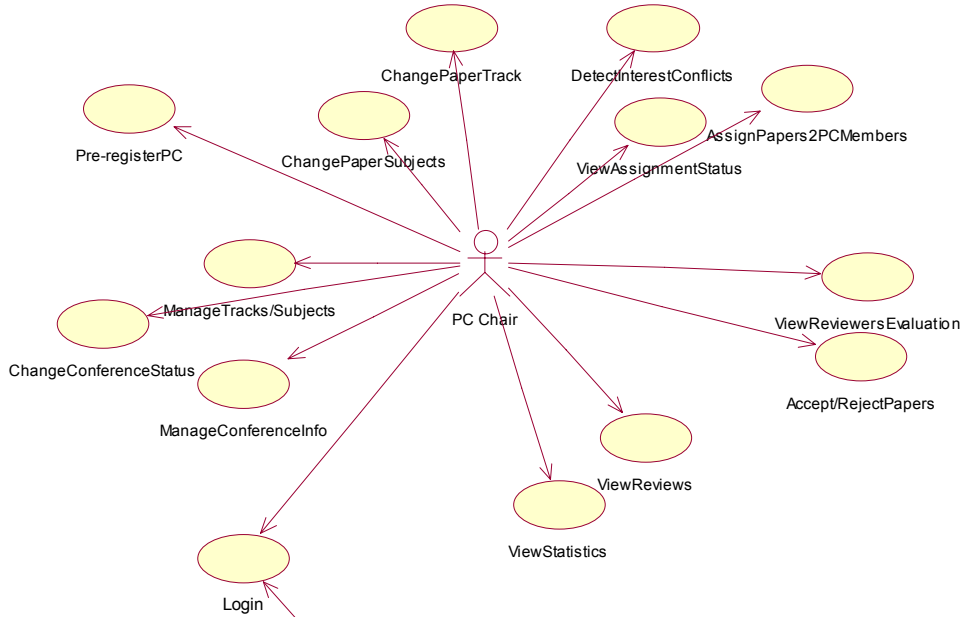

Fig 1: PCChair's Use Case Diagram

2.2. PCMember

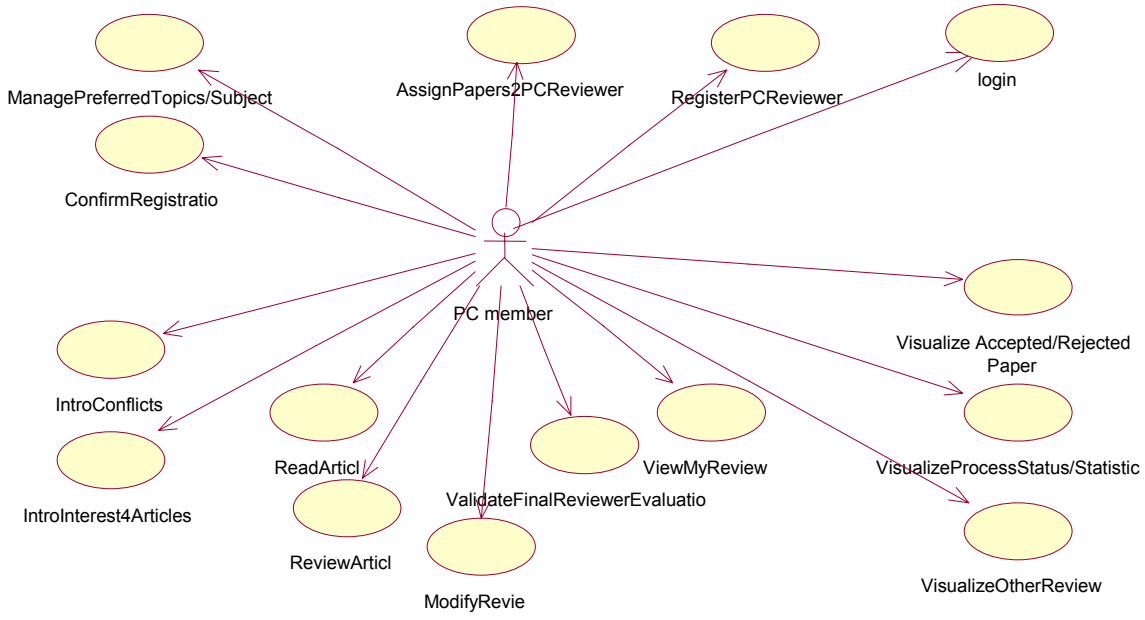

Fig 2: PCMember's Use Case Diagram

2.3. Reviewer

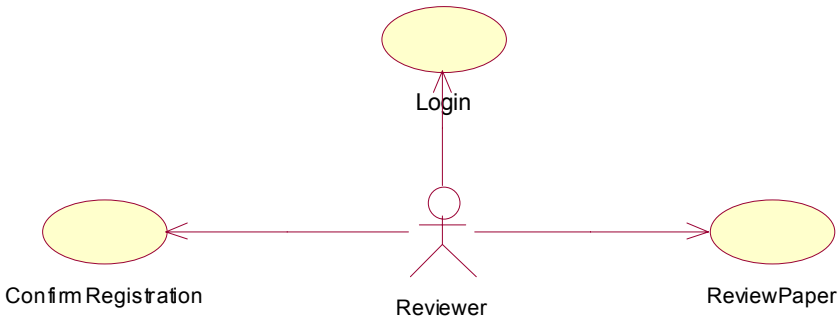

Fig. 3: Reviewer's Use Case Diagram

2.4. Author

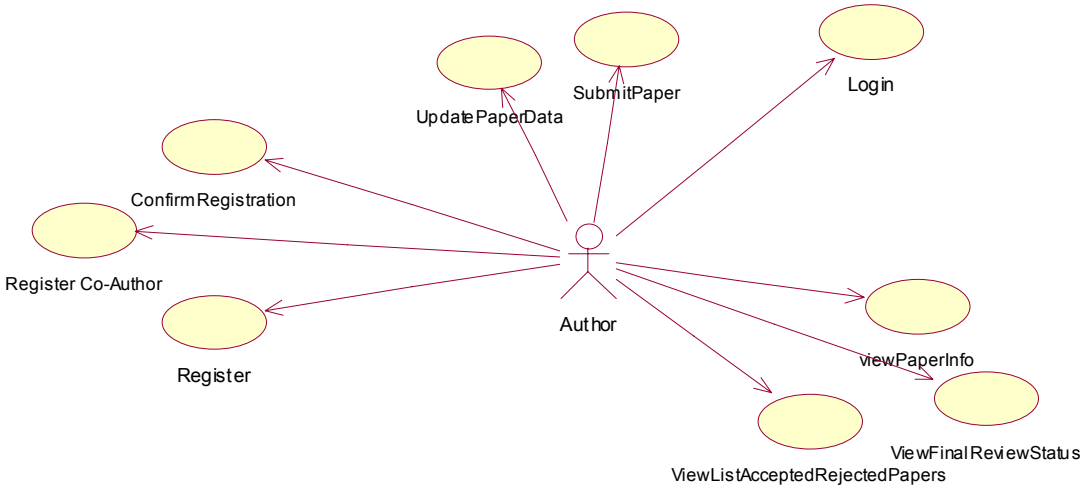

Fig. 4: Author's Use Case Diagram

Note that we have splitted the Use Case Diagram according to the different user types (Fig. 1 to 4) to more clearly separate the responsibilities (functional requirements) each actor has inside the boundaries of the system. In these diagrams (Fig. 1 to 4) use cases with the same name imply a functionality reuse. OO-H achieves that reuse degree by means of the use of Lt (Traversal Links, see section 1). It is also important to note that the use cases represented in OO-H are 'business use cases' and so their final implementation might vary depending on the target technology, architecture and/or target platform.

3.- Problem Domain

3.1. Analysis Class Diagram

OO-H departs from a UML standard class diagram. Just to clarify the Class Diagram presented in Fig. 5, note that a slash (/) next to an attribute/method stands for 'derived'. A dollar symbol ($) next to an attribute/method name stands for 'class-scope'[1]. Also, the <<*enumeration*>> stereotype defined on classes implies an enumerated type, with their attributes being the possible values of such type[2].

Our class diagram (see Fig. 5) depicts the following domain concepts (analysis classes): users (categorized into Authors, Reviewers, PCMembers and PCChairs), articles, revisions, tracks, and subjects. Moreover, we have detected two association classes: reviews (evaluation and comments a PCMember introduces in the system regarding a given paper) and revision preferences (interest degree and interest conflicts a PCMember has with reference to a given paper). Some attributes and methods have been directly derived from the description of the system. Others have been inferred from the application domain. The responsibility assignment has been realized taking into account which class was responsible for the most of the data involved in each one of the methods. Although we have tried to avoid method duplication in order to simplify the diagram, sometimes (mainly when we are dealing with methods that involve creation/deletion of relationships between objects) it might be convenient to provide the user with access modes from each one of the objects involved.

Based on the system description, we have identified six 'phases' for the paper revision process, all of them controlled by the PCChair. Each phase determines the functionality accessed by the different profiles.

- *AuthorSendingPapers*: In order to begin the revision process, the PCChair must introduce the conference parameters (PCChair data, conference dates, URL's, tracks, subjects, PCMembers involved in the revision process and so on) and, as the last step, open the period for the authors to submit papers. The system transition to this status might imply the sending of a '*Call for Papers*' to a set of selected distribution lists (DBWORLD, ISWORLD, etc).

- *PCChairIntroducingConflicts*: Once the paper submission period has expired (*conference.paperSubmissionDL*), the PCChair must change the status of the system in order to open the access to a new set of tasks. For example, in this new state the PCChair might revise the submitted papers, change their track and/or subjects if necessary, or look for revision conflicts (e.g. PCMember that are authors of a submitted paper).

[1] The class-scope $ notation has been deprecated and substituted by an underlined attribute/method name in the last versions of UML. OO-H will change the notation support accordingly in future versions of the tool.

[2] This way of defining enumerated types has been included in UML 1.4

- *PCMemberIntroducingPreferences*: The following step is to open the system for the PCMembers to introduce their preferences regarding the submitted papers, as well as revision conflicts not detected by the PCChair (if any).

- *PCChairAssigningReviews*: Once the PCChair closes the period to register paper preferences, and taking into account the preferences and conflicts registered in the system, it is time for the PCChair to assign papers to the different PCMembers for revision.

- *PCMemberReviewingPapers*: Then, the period for each PCMember to introduce its revisions is opened. Again, the system transition to this state implies the sending of an e-mail to each PCMember with information regarding the papers s/he has been assigned and the period of time s/he has to perform the revision. This phase ends when the review deadline (*conference.reviewDL*) is reached.

- *PCChairEvaluatingPapers*: Once the PCChair sets the state of the system to this value, and taking into account the reviews introduced by the PCMembers (or the corresponding external reviewers), he must decide which papers are accepted and which ones rejected.

- *ProcessFinished*: This last step implies the sending of an e-mail to all the paper authors, informing them of the revision process result regarding their papers.

OO-H considers the e-mails the system must send on some conference state transitions are isolated inside the body of the c*onference.changeProcessStatus()* method, and so out of the scope of our models.

3.2. Modelling Assumptions

In order to simplify the diagrams, we have applied the <<*singleton*>> pattern [2] to the *'Conference'* class. This pattern implies that the generated system deals with a single conference (i.e. there may exist just one object of type *'conference')*. All tracks, subjects and people in the database are implicitly related to that conference. Extending such system to deal with several conferences at a time is trivial.

Furthermore, we consider the PCChair cannot revise articles nor delegate the paper revisions to external reviewers. He cannot submit any paper (as author) either. Also, a PCChair cannot assign a paper to a PCMember that is not interested (interest degree==0) or that has a conflict with such paper. Furthermore, in our system a PCMember can express the same interest degree for several articles (i.e. the interest degree doesn't imply a 'strict preference order'). Also, an external reviewer cannot be assigned more than once to the same paper. Each paper can be updated by any of its authors as long as the submitting period has not expired. A given paper is associated to a single track, but it can be associated to any number of subjects.

It is also convenient to note that some attributes and methods, as well as the mandatory/non mandatory character and default value of attributes and method parameters, have been specified following a personal criterion, as nothing is stated in the description regarding such aspects.

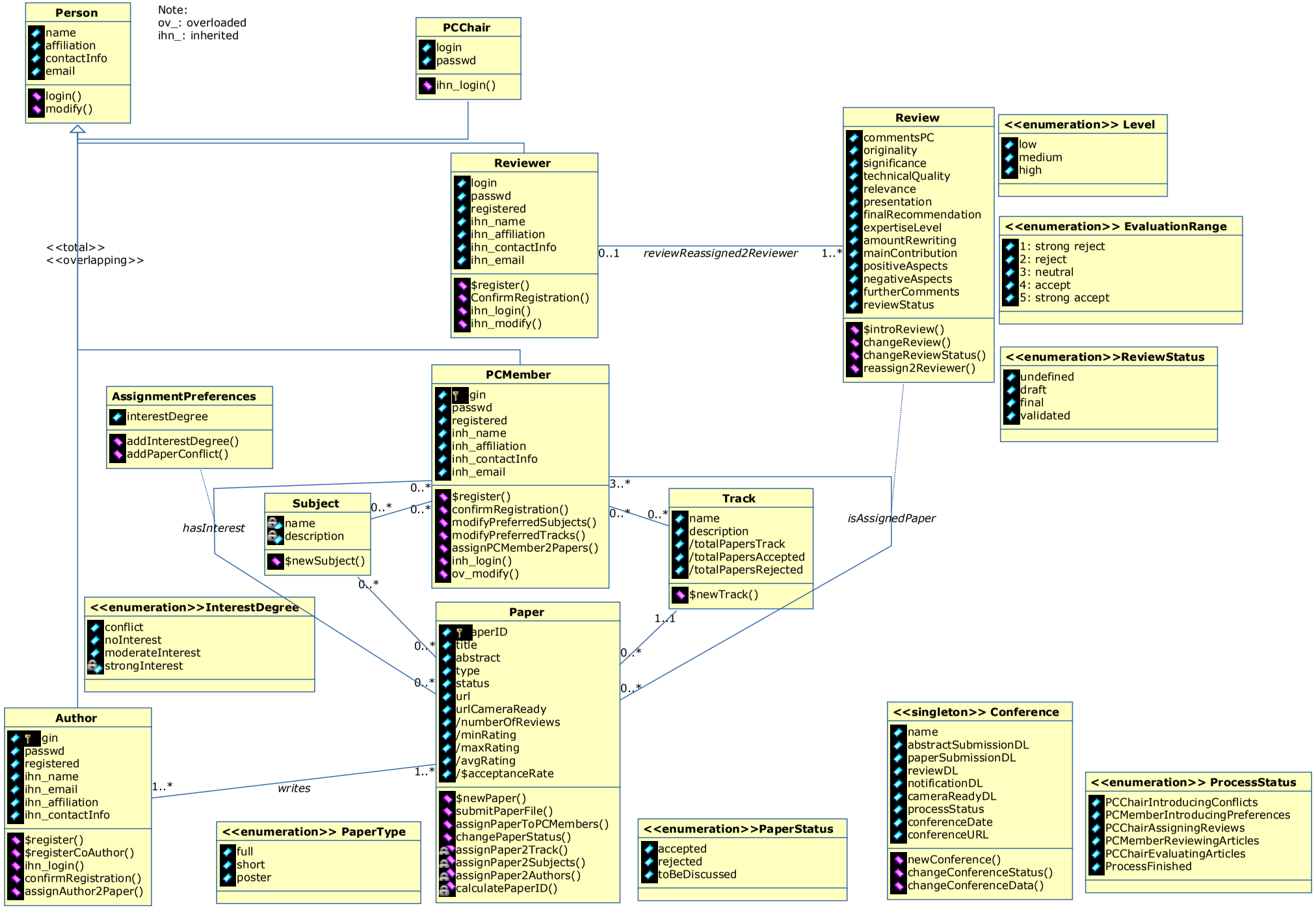

Fig. 5: The Conference Review System Class Diagram

4.- Navigation modelling

4.1. Introduction

The construction process of the navigation access diagrams (NAD) is divided into three steps, namely:

- Construction of the Use Case diagram
- Association of a storyboard (mockup of the interface) to the different Use Cases to better illustrate user-system interaction.
- Construction of the NAD diagrams that model such storyboard and provide access to the methods needed to fulfill the Use Case functional requirements.

These steps are not necessarily sequential. Furthermore the different views (use cases, storyboard, NAD's) can be further refined in new iterations.

Designers usually depart from a storyboard that complements a use-case diagram and gathers the idea he has about what the application should be about and so makes for a 'contract' with the end user. It is this idea what drives the rest of the process. In OO-H both artifacts help the designer to decompose the system interface into subsystems and pages that clearly fulfill a set of functionality and navigation requirements.

However, in order to better illustrate how the NAD's are constructed, in this article we have followed a different approach: (1) first, we present the Use Case diagrams showed in section 2. We have grouped the use cases attending different criteria[3], and we have depicted these grouping decision by means of package symbols around the use cases involved. Note that these 'packaged use-cases' are not part of the model, and are depicted just to show the Navigation Target in which each Use Case interface is modelled. In this way we make sure the whole interface is captured at NAD level. (2) Then we show how this grouping process acts as a starting point from where the level 0 NAD diagram may be derived. (3) Last, we illustrate, by means of the storyboard corresponding to the 'Manage Conference' Navigation Target, the mapping of the different NAD abstractions into abstract pages and constructs.

Next, we are presenting the NAD's construction process.

4.2. Navigation Access Diagrams

In this section we will present, step by step, the construction process of the NAD diagram corresponding to the PCChair. The process followed to construct the diagrams corresponding to the other actors (PCMembers, Reviewers and Authors) is analogous.

[3] These grouping criteria will be further explained in section 4.2.1

4.2.1. PCChair Navigation Profile

If we look again at the PCChair use-case diagram (see Fig. 1) we will observe the set of functional requirements her interface must fulfill.

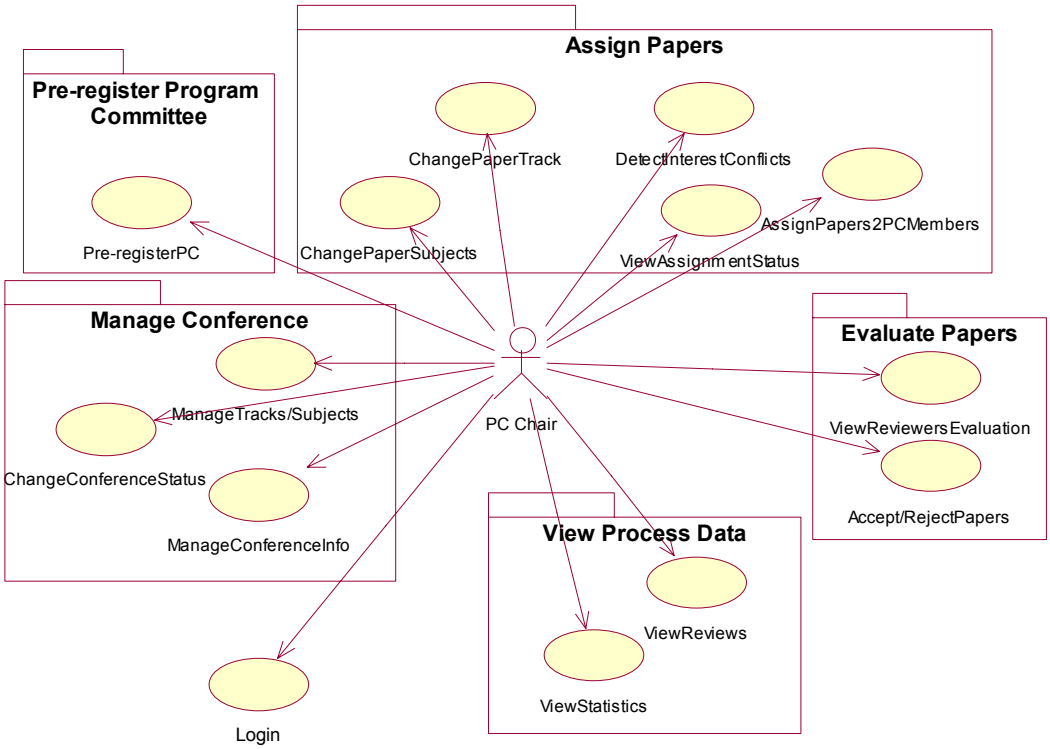

Fig. 6: PCChair's Use Cases grouping process

In section 2 we also commented how this diagram was useful to decide how to group those requirements into 'Navigation Targets', attending at either semantic, functional dependency, data considerations or a mixture of them. We say we are using 'semantic criteria' when we group use cases that have a similar aim. As an example, in Fig. 6 we can observe how the use-cases '*view Reviews*' and '*view Statistics*' have been grouped under the NT '*View Process Data*', due to the fact that both provide reports on the review information (one aggregated, the other detailed) contained in the system. On the other hand, in order to gather '*view Reviewers Evaluation*' and '*Accept/Reject Papers*' we have applied what we call a 'functional dependency' criterion, that is, we have departed from the premise that, in order to be able to accept/reject papers, we must have a synthesized view of every reviewer evaluation that helped the PCChair to take a decision. Last, the use-cases '*ChangeConferenceStatus*' and '*ManageConferenceInfo*' have been grouped under the NT '*Manage Conference*' following a data criterion, that is, due to the fact that both access and manipulate data of the '*Conference*' class.

The grouping process induces an interface structure. Consequently, the more careful this process is performed, the higher the quality of the final interface structure will be.

Next we are going to show the NAD diagrams corresponding to the first actor of the system, the PCChair. Together with them, we are showing the *storyboard*

corresponding to the first NT, *'Manage Conference'* in order to illustrate the mapping process.

PCCHAIR PROFILE ENTRY POINT

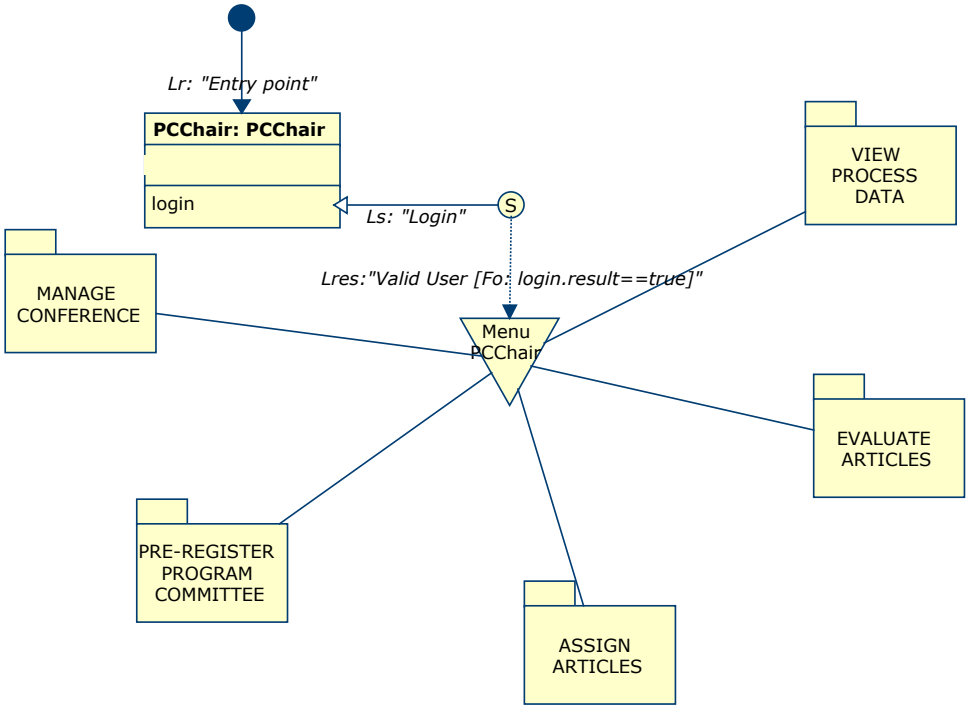

Fig. 7: PCChair NAD. Level 0

In Fig. 7 we observe the modelling of the entry point to the application (represented by the requirement link *'Entry Point'*). One possible set of storyboard pages corresponding to this diagram is showed in Fig. S1 and S2. The first abstract page corresponds to a form for the user to log in the system. This process involves a user login, passwd and profile, which correspond to the parameters (all of them mandatory) of the method *PCChair.login()*. If the user exists (condition that is reflected in the Fo '*Valid User*'), s/he will be showed a menu where a link to each of the five NT identified (see Fig. 7) is presented. Such menu is represented by the collection construct *'Menu PCChair'*.

Next, we are showing the internal structure of each NT, and the first of them, '*Manage Conference*', is described in detail.

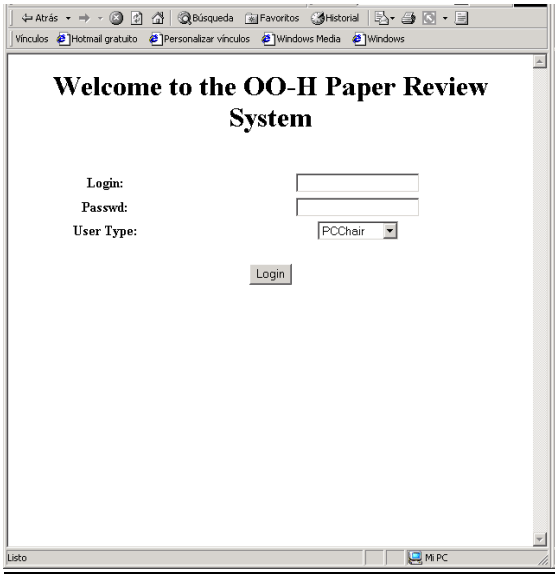

Fig. S1: Login

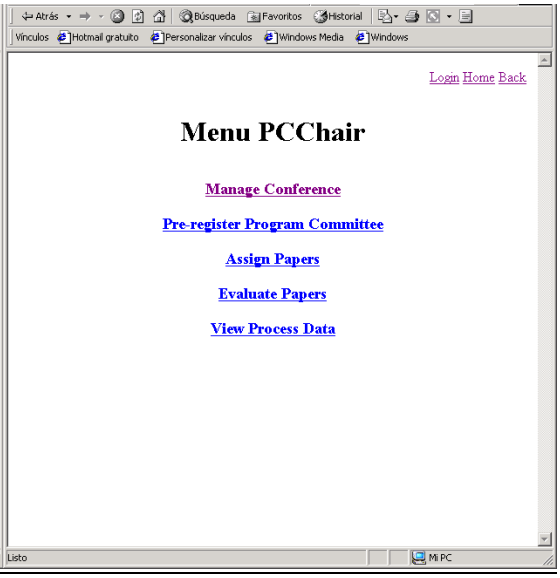

Fig. S2: Menu PCChair

NAVIGATION TARGET 'MANAGE CONFERENCE'[4]

When the PCChair selects the *'Manage Conference'* option, it access the interface subsystem modelled incide the corresponding NT. The entry point to this NT (requirement link '*Conference Maintenance'*, see Fig. 8) points at a new collection, called '*Conference Menu'*, that differs from the previous one in the type of links that depart from it. While in the '*PCChair Menu*' collection the links had the visualization attribute set to *'show in destination',* this time they are of type *'show in origin'*. That means that, in this case, no link to a new page is generated, but that the information corresponding to the destination classes is directly presented to the user, provided that the corresponding Fo is evaluated to true. Furthermore, as both filters (conference.population()==0 / conference.population()>0) are disjoint, only one view will be available at a time: if the conference hasn't been created yet (it is the first time the PCChair enters the system) the screen corresponding to the '*createConference()*' method will appear (see Fig. S3). This page gathers the set of parameters the method '*New Conference*' requires. Once the method has been invoked and the control returned to the interface, the response link '*Conference Created*' drives the user again to the *'Conference Menu'*. This time, however, when the filters are checked again, it is the *'View Conference'* link the one that is evaluated to true, and so the system will automatically generate the page shown in Fig. S4. This page provides a view of the conference data, together with access to different maintenance options for the conference.

[4] In the OO-H diagrams, an asterisk next to the link name means that the set of activation links is not complete (that is, is not made up by every link from which the user might have arrived to the actual view). Also, an arrow with a filled head means that its visualization metamodel attribute is set to '*show in destination*', whilst, if it has a hollow head, the corresponding value is *'show in origin'*.

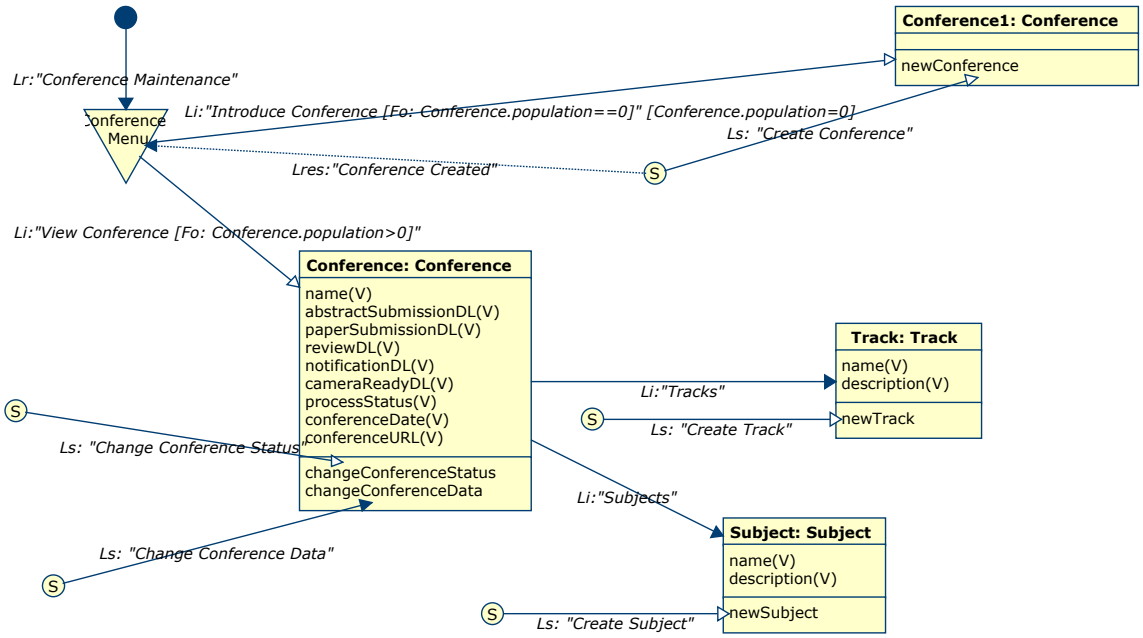

Fig. 8: Manage Conference

In Fig. S4 we can also observe how, due to the fact that the service link '*Change Conference Status*' has its visibility attribute set to '*show in origin*', the form corresponding to the method invocation (with a list of the possible conference status, due to the 'selection' introduction mode associated to the parameter[5]) appears together with the conference information view. The two Internal Links and the Service Link defined with the attribute visualization='show in destination' generate three links (in this case represented by buttons) to the views "*Change Conference Data*" (see Fig. S5), *'Tracks'* (see Fig. S6) and *'Subjects'* (see Fig. S7). In Fig. S5 we can also observe how the different method parameters may have a default value associated (in this case the actual value of the corresponding class attributes). Also note that, when not otherwise stated, the Response Link comes back to the view that contained the service.

As the reader will have already inferred, is the visualization attribute what characterizes the final abstract page structure of the interface. We call this page structure abstract because there is nothing that prevents those pages to be further composed into a frame structure or any other mechanism that allows the coexistence of different views of the system on the same physical screen. In Fig. 9 we can observe the interconnection of the screenshots captured in Fig. S1 to S7. In this figure, the page separation is determined by the position of the destination links. In section 5 we will explain how this siteview interconnection perfectly matches with the APD diagram generated by the OO-H CASE tool departing from the corresponding NAD diagram. Also, note how the storyboard reflects the fact that OO-H automatically generates, if not otherwise stated, a link from every abstract page to the NT origin, another one to the application entry point and another one pointing at the previous abstract page in the navigation path.

[5] See Section 1 for a description of the different introduction modes for method parameters

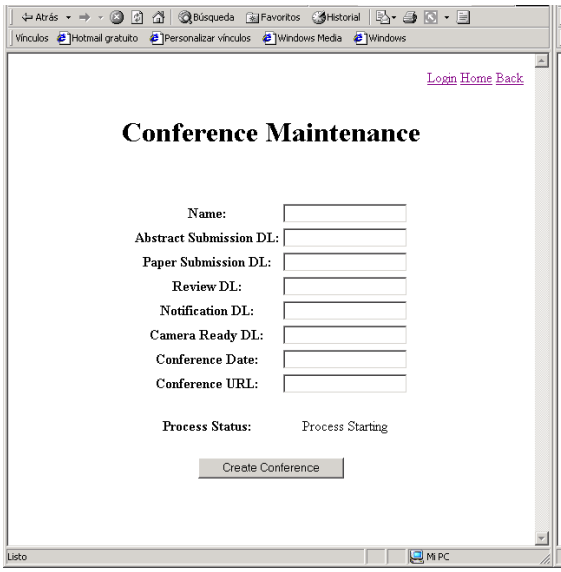

Fig. S3: New Conference

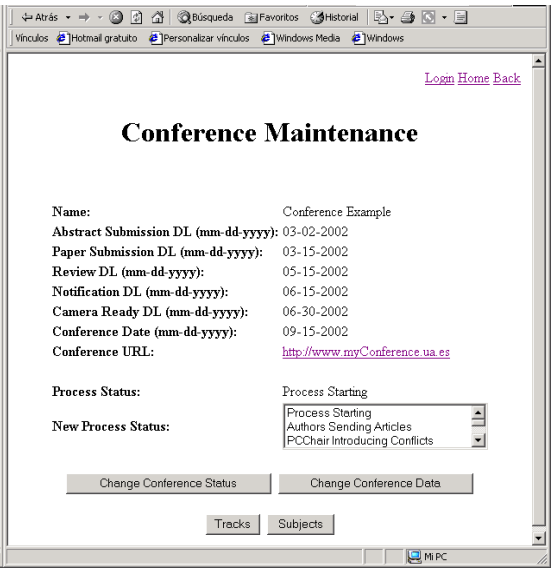

Fig. S4: Conference View/Change Status

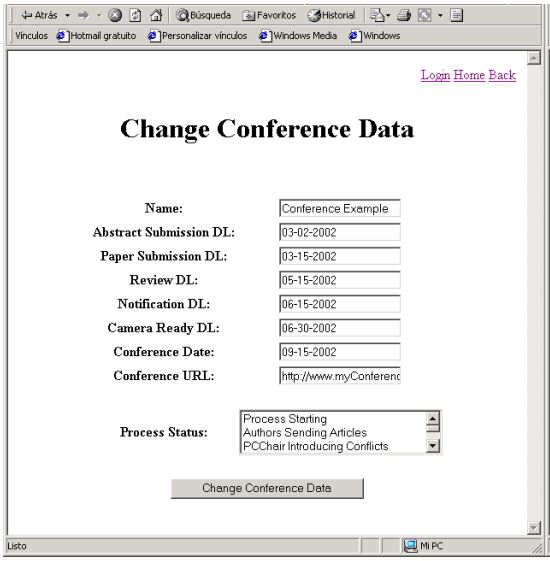

Fig. S5: Change Conference Data

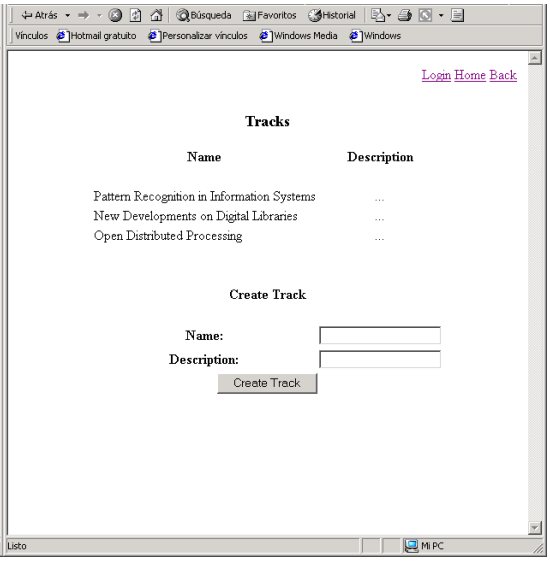

Fig. S6: New Track

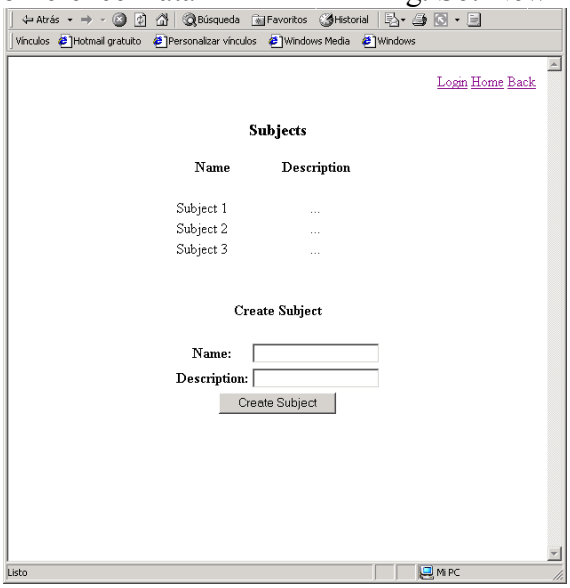

Fig. S7: New Subject

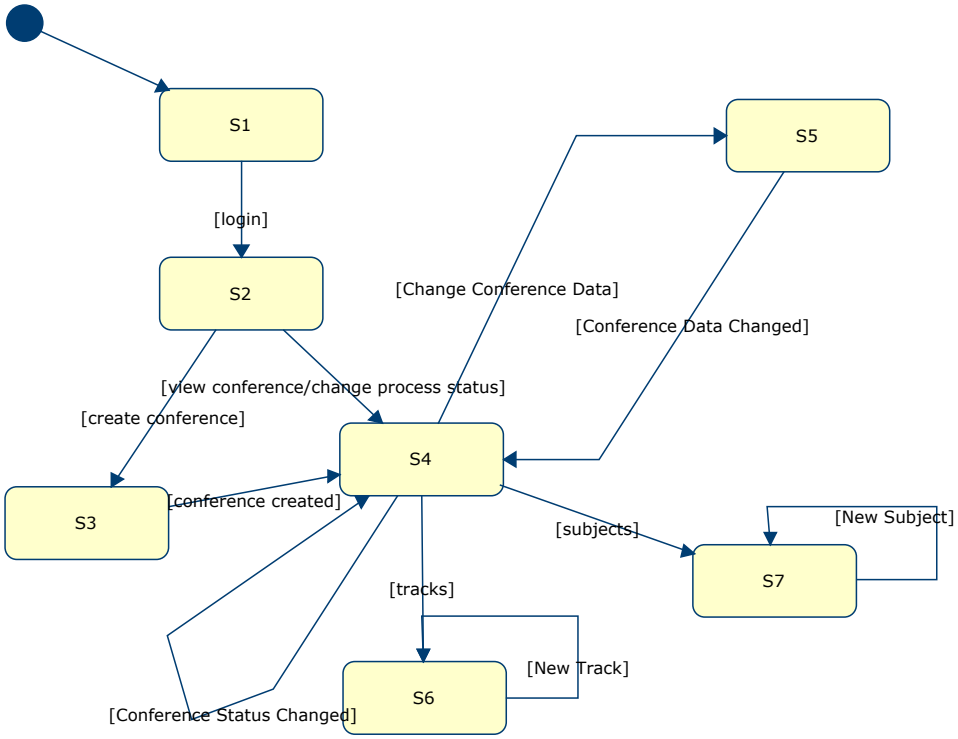

Fig. 9: Siteview corresponding to the storyboard shown in S1 .. S7

In the remaining of the section, we are showing the other NT related to the PCChair, with a brief description of their main features.

NAVIGATION TARGET 'PRE-REGISTER PROGRAM COMMITTEE'

In this NT the PCChair obtains a view of every PCMember that has been registered in the system, together with the form that gathers the parameters and invokes the *PCMember.register()* method, necessary to add new PCMembers to the system.

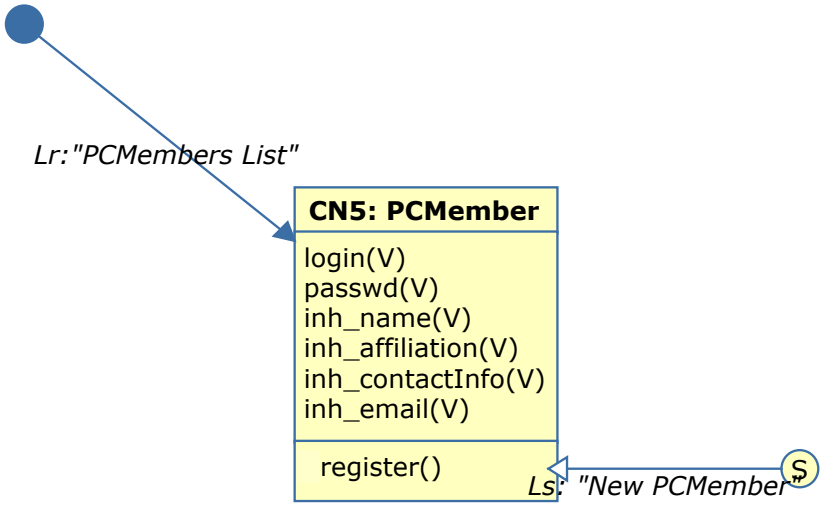

Fig. 10: Pre-register Program Committee

NAVIGATION TARGET 'ASSIGN PAPERS'

In this NT (see Fig. 11) the PCChair accesses a menu with two options: '*Assign by Paper*' and '*Assign by PCMember*'. In case we chose the first one, we will access a page where every submitted paper appears. For each paper the interface provides three possibilities:

- Assign a new track to the selected paper
- Assign new Subjects to the selecte paper
- Assign the review of the paper to a PCMember

Although the way the PCChair changes the track and/or subjects associated to a paper is immediate, the way it selects the PCMembers more suitable for the paper revision is not that trivial. The associated '*PCMemberList*' parameter corresponding to the method '*Paper.AssignPapers2PCMember()*' has a navigation introduction mode[6]. The navigation diagram capturing the view required to choose the '*PCMemberList*' population is shown in Fig. 12. This diagram corresponds to the explosion of the '*SelectPCMembers*' NT represented in Fig. 11 and associated to the method that owns the '*PCMemberList'* parameter by a dependency arrow.

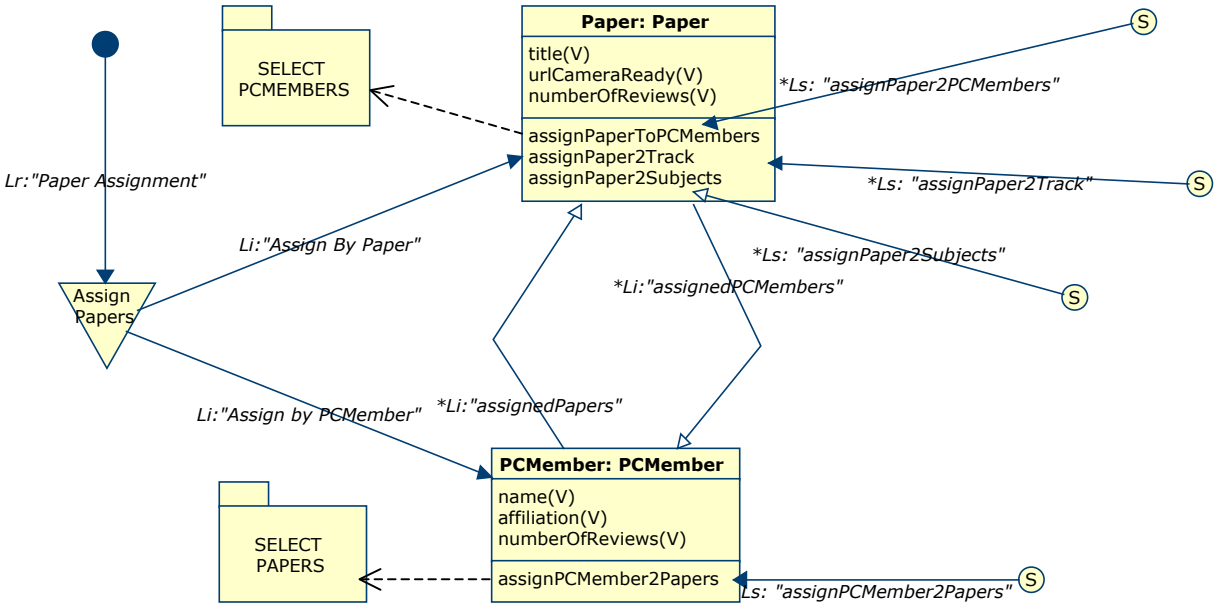

Fig 11: Assign Articles 2 PCMembers

Briefly, in order to select a set of PCMembers to review a given article we have modelled a view in which the relevant paper data (track, subjects, title) is shown together with the PCMembers that have no conflict with the actual paper and their corresponding interest degree in the paper, the id of the papers that each PCMember has already assigned for review, her preferred tracks and preferred subjects. With this information the PCChair is able to decide who are the ideal candidates to revise the paper. The fact that we can select a set of PCMembers at a time is modeled by the metamodel attribute 'application scope' set to 'Multiple' in the '*Reviewer Selected'* exit

[6] See section 1 for a further discussion of the different parameter introduction modes.

link. When we reach such exit point, the selected PCMembers are returned to the method as the *'PCMemberList'* parameter.

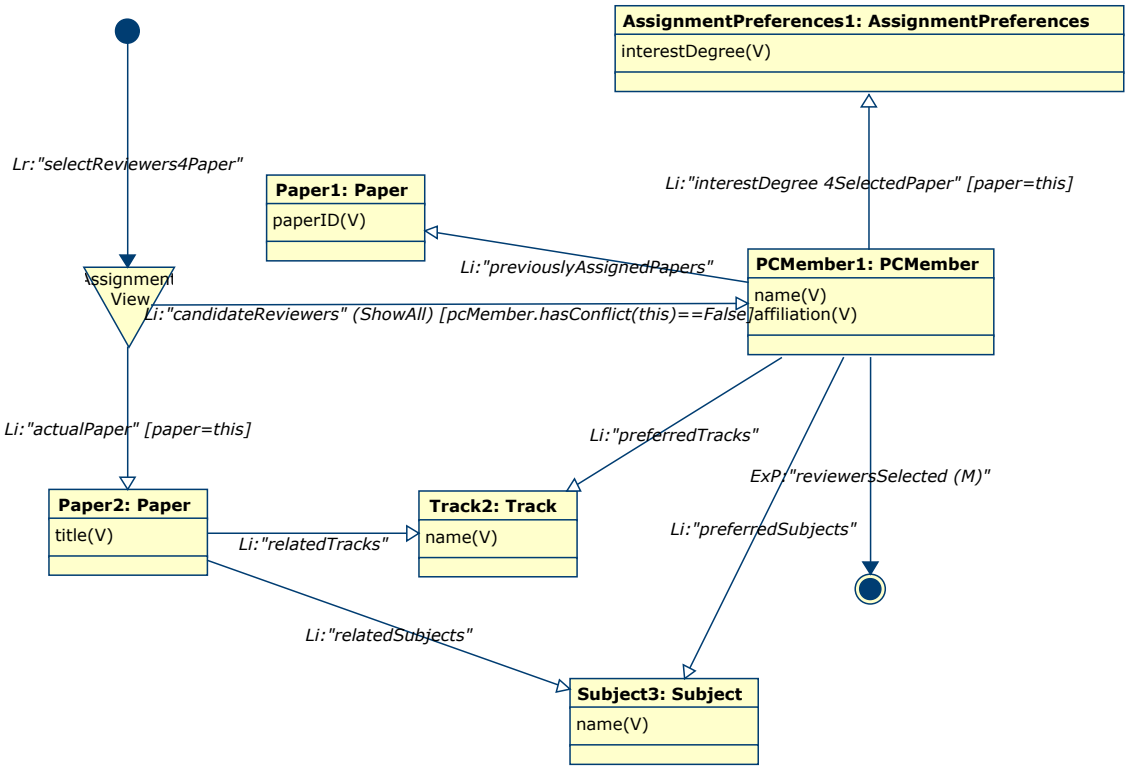

Fig. 12: 'Select PCMembers' NT

Also in Fig. 11 we can observe the effect of the 'activation link' concept, explained in section 1. Supposing that the designer has decided that the paper-related methods are only available when we have arrived to this class through the '*Assign By Paper*' link, we must eliminate the '*assignedPapers*' from its set of activation links.

The second possibility in Fig. 11 (*'Assign by PCMember'*), implies the generation of a page that contains the list of PCMembers, together with the number of revisions they have already assigned. This view also contains the possibility of assigning new articles to a given PCMember. The model corresponding to the NT '*Select Papers*', that gathers the way the '*paperList*' parameter corresponding to the P*CMember.assignPCMember2Paper()'* is introduced can be seen in Fig. 13.

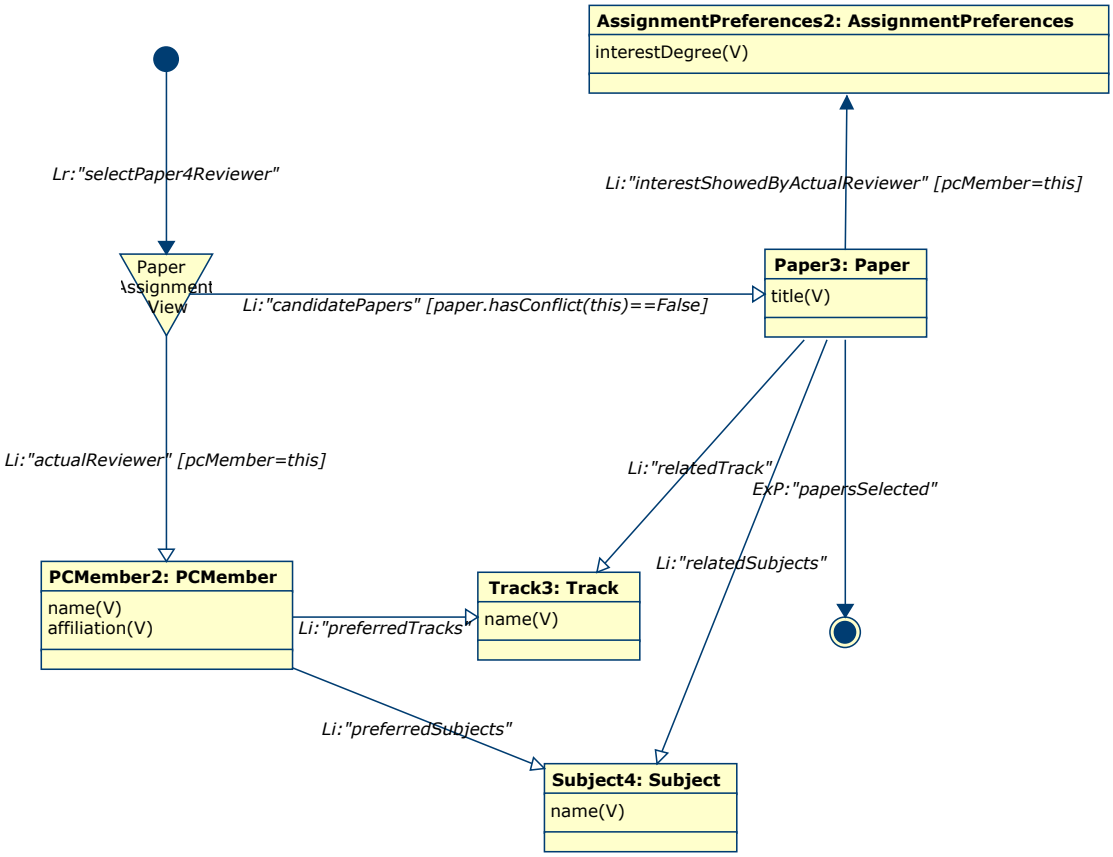

Fig. 13: 'Select Papers' NT

NAVIGATION TARGET 'EVALUATE PAPERS'

This NT captures two different kinds of report: list of papers ordered by ID, and list of papers ordered by average rating (*paper.avgRating*). Both reports show the same information items: minimum score, maximum score and average score, login of each PCMember that have reviewed the paper and final recommendation of each one of them. Moreover, the PCChair can, in this view, decide whether s/he accepts/rejects each paper.

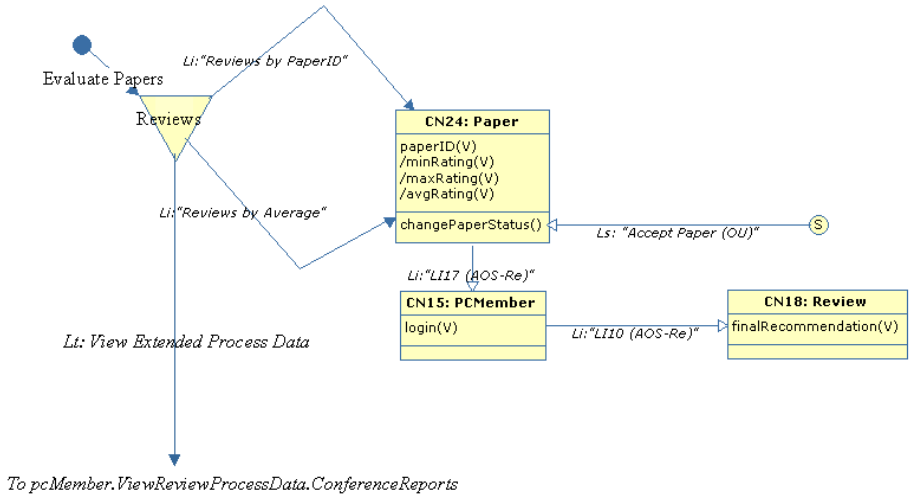

Fig. 14: Evaluate Papers

Note how this NT reuses, by means of a Traversal Link, the views modelled in the NT '*PCMember.View Review Process Data'*, that will be introduced in the following section.

NAVIGATION TARGET 'VIEW PROCESS DATA'

In the views modelled by this NT the PCChair obtains all the data regarding papers with its corresponding reviews and track statistics. Note also the use of a Traversal Link to share the views defined in the '*PCMember.ViewReviewProcessData'* NT.

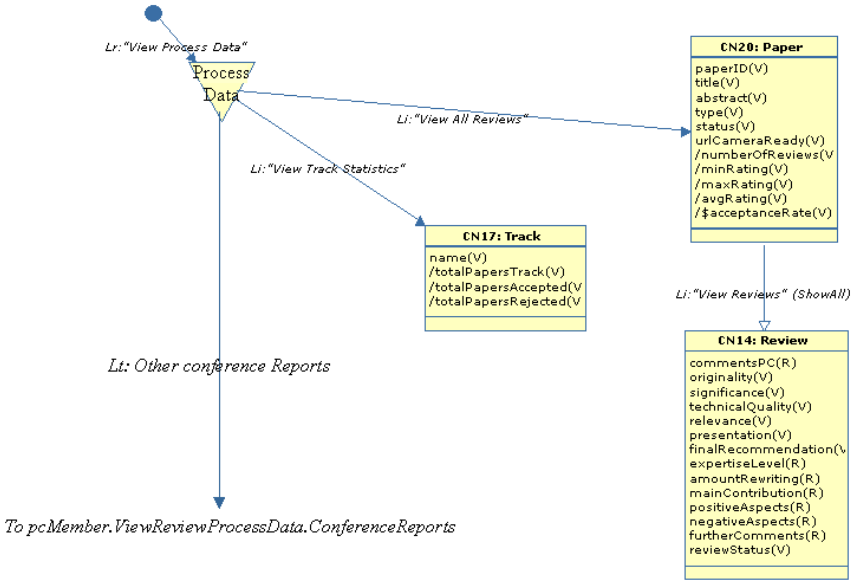

Fig. 15: View Process Data

4.2.2. PCMember Navigation Profile

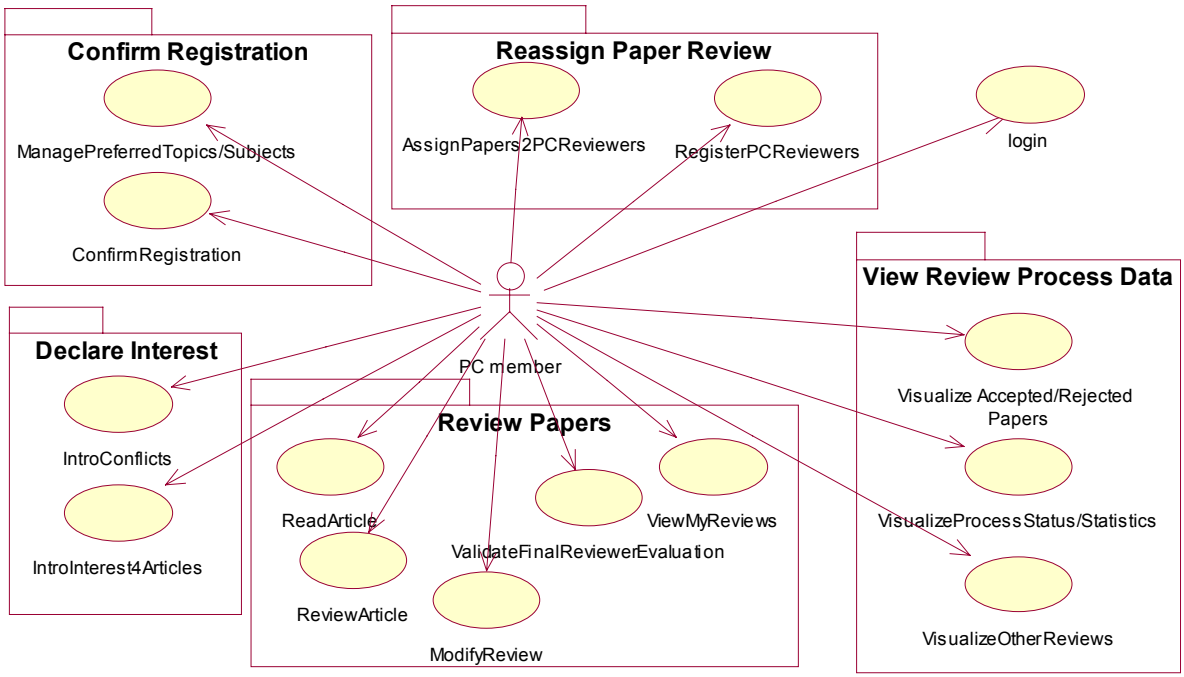

Fig. 16: PCMember's Use Cases grouping process

PCMEMBER PROFILE ENTRY POINT

The entry point to the PCMember profile is analogous to that presented for the PCChair.

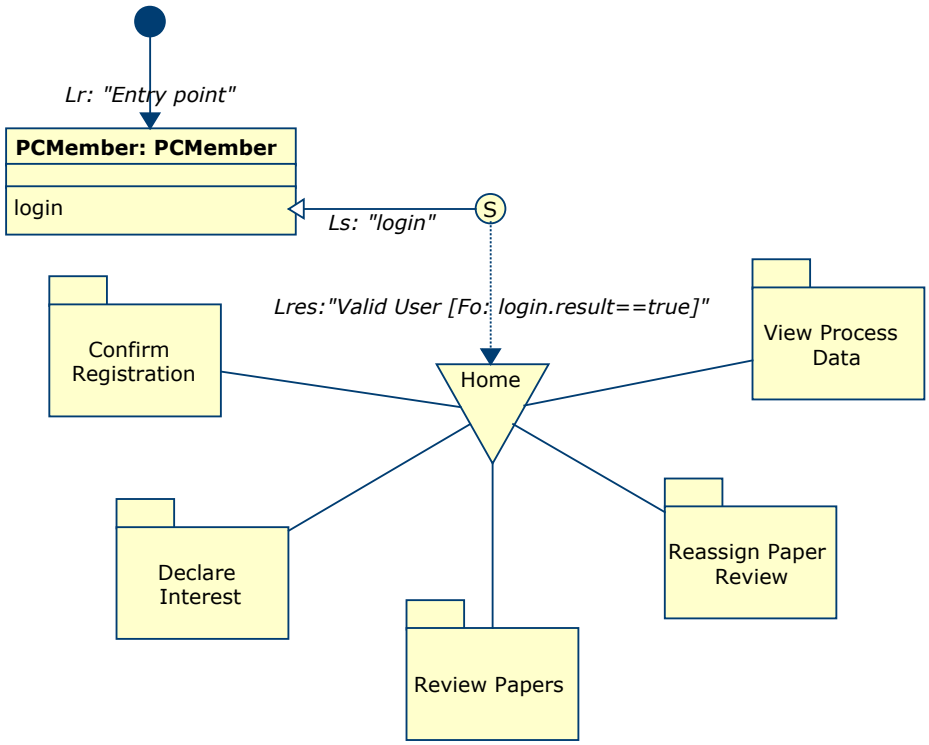

Fig. 17: PCMember NAD. Level 0

NAVIGATION TARGET 'CONFIRM REGISTRATION'

Every PCMember must confirm the data introduced by the PCChair before gaining access to the system. Also, s/he must select the tracks/subjects s/he masters.

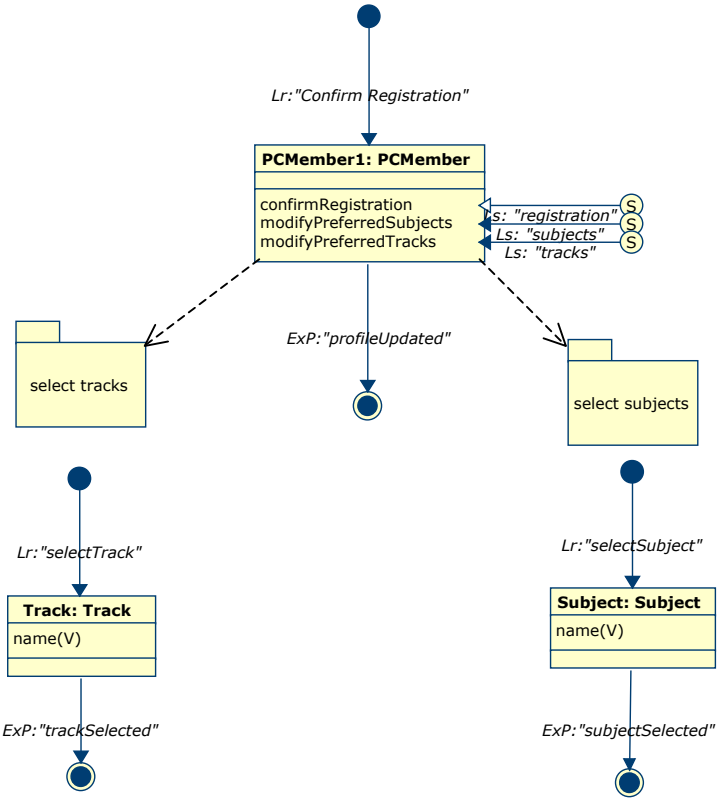

Fig. 18: Confirm Registration

NAVIGATION TARGET 'DECLARE INTEREST'

Once the PCMember has confirmed her registration, and during the period the PCChair establishes for introducing preferences about papers to review, the PCMember can access a page where, for each paper, and looking at its title, abstract, associated track and subjects, s/he can introduce her interest degree. Note that, for every Internal Link, we have activated the underlying structural relationship (Re) that acts as Fd for that link[7]. Also, all of them are automatic, simple (that is, tracks, authors and subjects are presented for each paper) and visualized in origin (together with the paper information).

[7] see section 1 for a broader explanation of filters

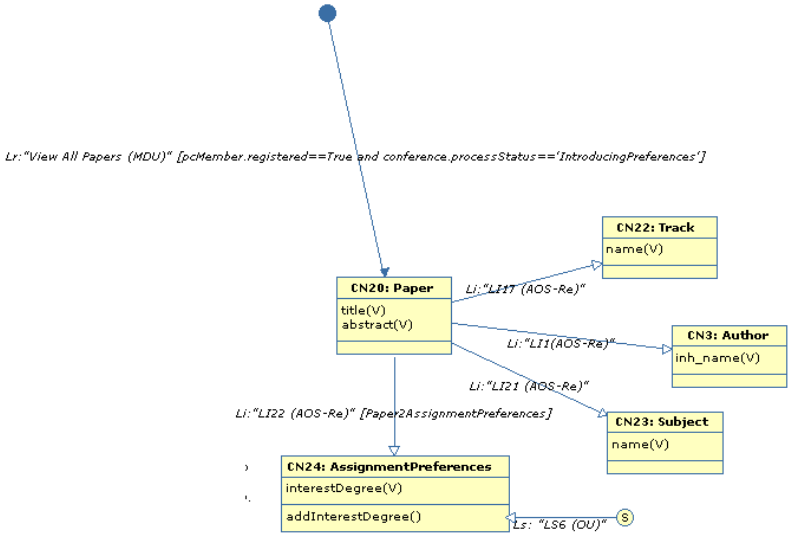

Fig. 19: Declare Interest

NAVIGATION TARGET 'REVIEW PAPERS'

This NT (see Fig. 20) models a page for the PCMember to view, introduce and modify either her or any of her external reviewers paper revisions. In this Nad the concept of activation link becomes crucial to capture the requirements concerning when can a Pcmember introduce/modify a review. Only if we are coming through 'ViewPapers2Review', 'View Uncompleted Reviews' or 'Validate Reviews' (introduced by an external reviewer) are we allowed to change the data concerning the paper review. Also, only if we are coming through the 'ViewPapers2Review' (which generates a view of the papers we haven't reviewed yet) are we allowed to introduce a new review or reassign the review to an external reviewer (Lt "Reassign Paper Review", which connects to the NT presented in Fig. 21).

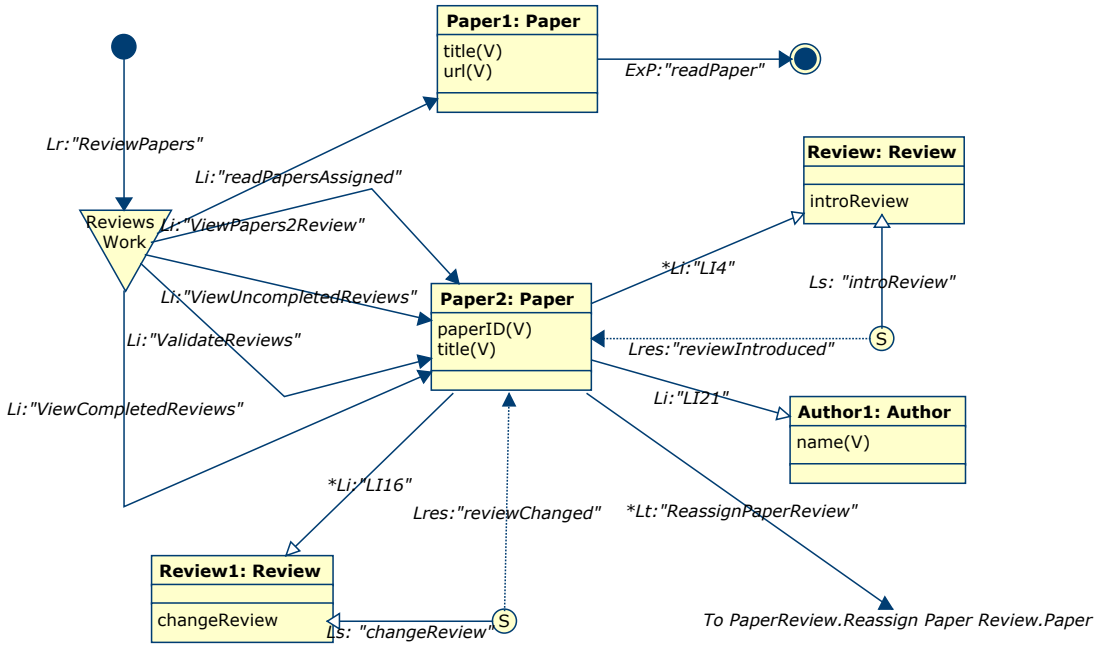

Fig. 20: Review Papers

NAVIGATION TARGET 'REASSIGN PAPER REVIEW'

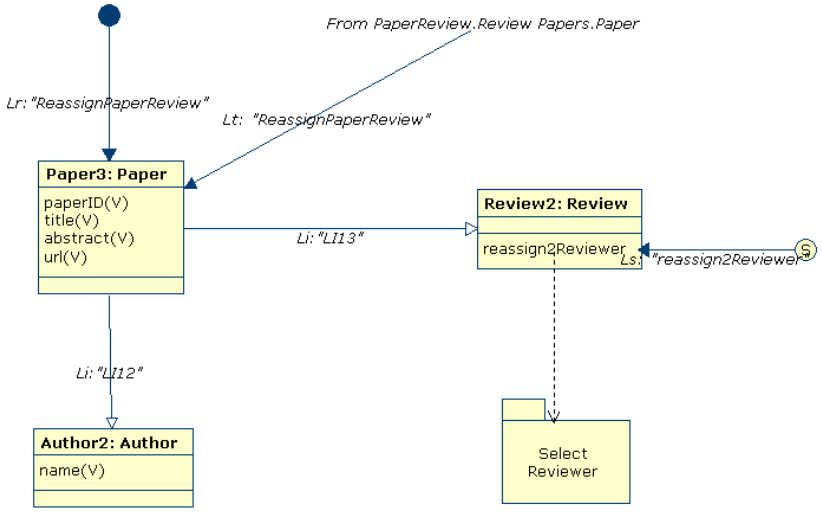

Fig. 21: Reassign Paper Review

NAVIGATION TARGET 'VIEW PROCESS DATA'

The PCMember can, at any time, view the final revision results (title of the paper, authors and acceptance status. Also, s/he can view statistics concerning general data per track and average acceptance percentage. While the PCChair is reviewing a paper, s/he cannot look at the reviews introduced by other reviewers. This constraint is captured in the *Fd: reviewStatus=='validated'* associated to the *'ViewMyPaperReviews'*. However, once the s/he has introduced her reviews and/or when the review deadline is reached, s/he can access the reviews any PCMember has introduced for any paper (providing s/he has no conflict).

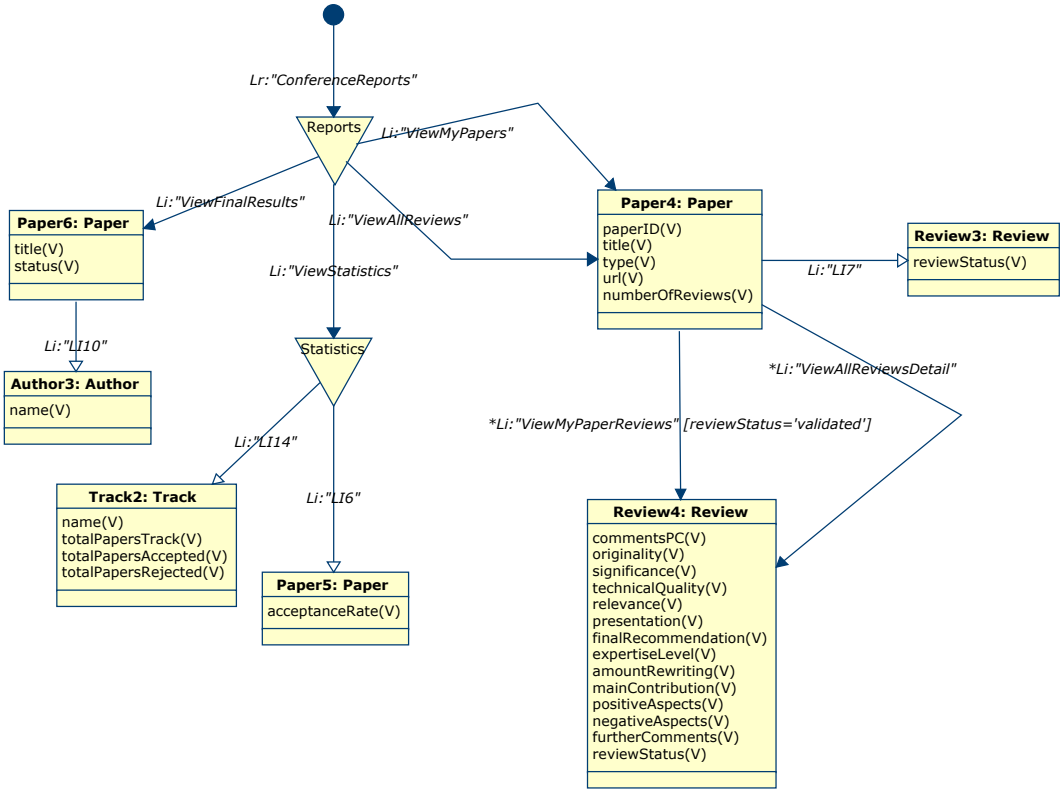

Fig. 22: View Process Data

4.2.3. External Reviewer Navigation Profile

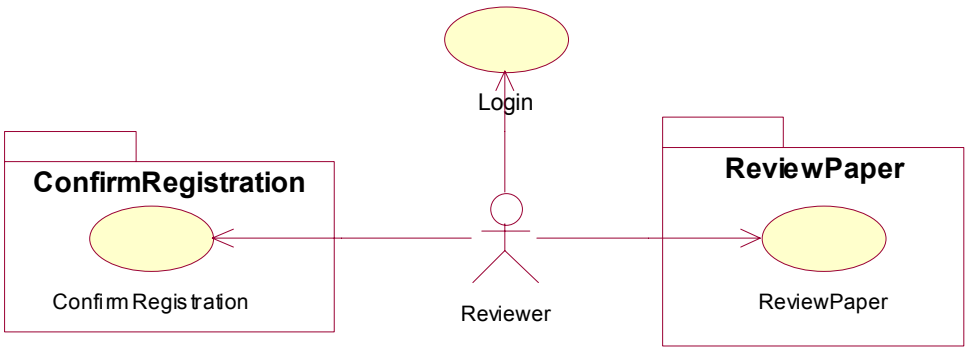

Fig. 23: Reviewer's Use Cases grouping process

In Fig. 23 we can observe the main responsibilities of an external reviewer: confirm her registration and review the papers assigned to him by a PCMember.

EXTERNAL REVIEWER PROFILE ENTRY POINT

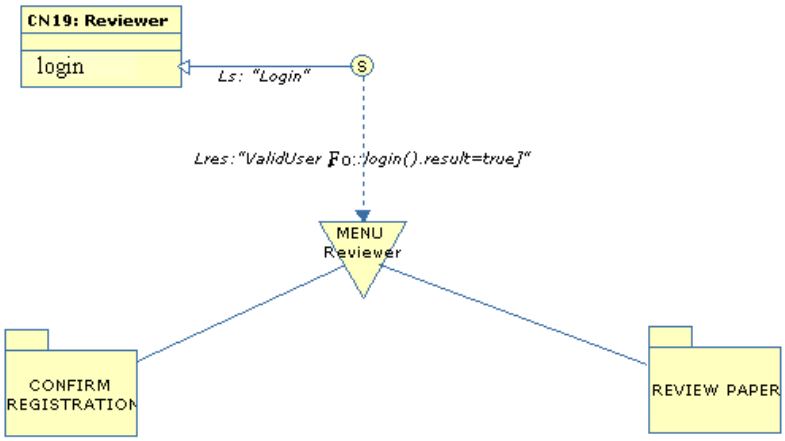

Fig. 24: External Reviewer NAD. Level 0

The entry point to the profile is similar to that presented for the rest of the profiles.

NAVIGATION TARGET ‘CONFIRM REGISTRATION’

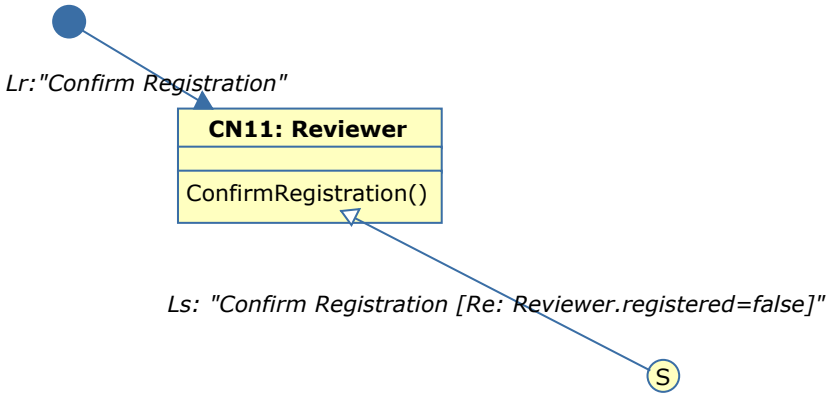

Fig. 25: Confirm Registration

This NT is quite straightforward, and allows the reviewer to validate her data (parameters of the method *reviewer.confirmRegistration()*)

DESTINO NAVEGACIONAL ‘REVIEW PAPER’

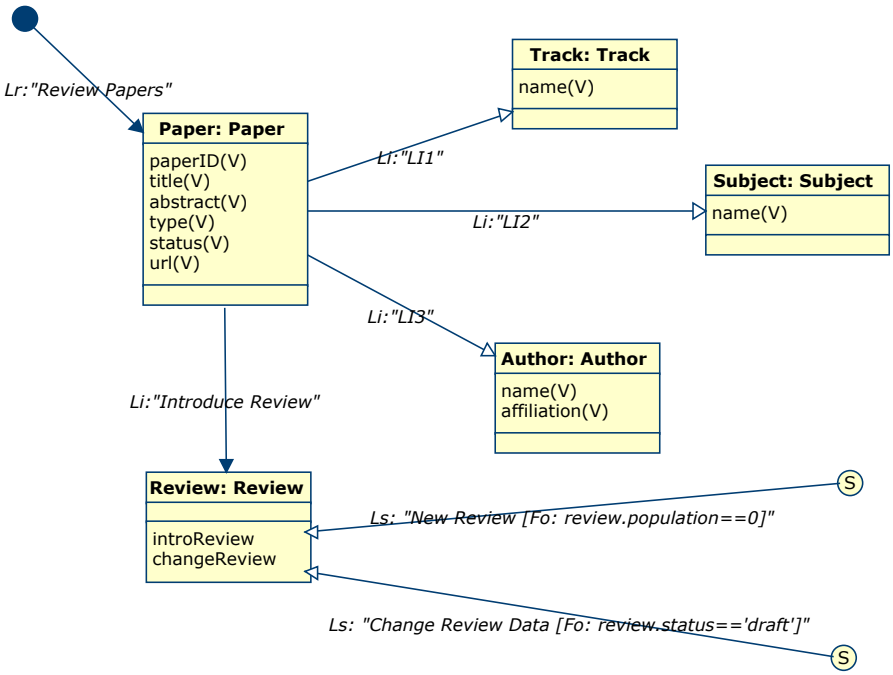

Fig. 26: Review Paper

In this NT (see Fig. 26) the view modelled presents the set of papers assigned to a given reviewer, together with their corresponding track, subjects and authors. The first time the reviewer reviews the paper (that is, there is no review associated to that paper with that reviewer) the *review.introReview()* method is activated. From then on, the reviewer is allowed to work on the revision (*review.Change Review Data()*) as long as this revision is not set to ‘final’.

4.2.4. Author Navigation Profile

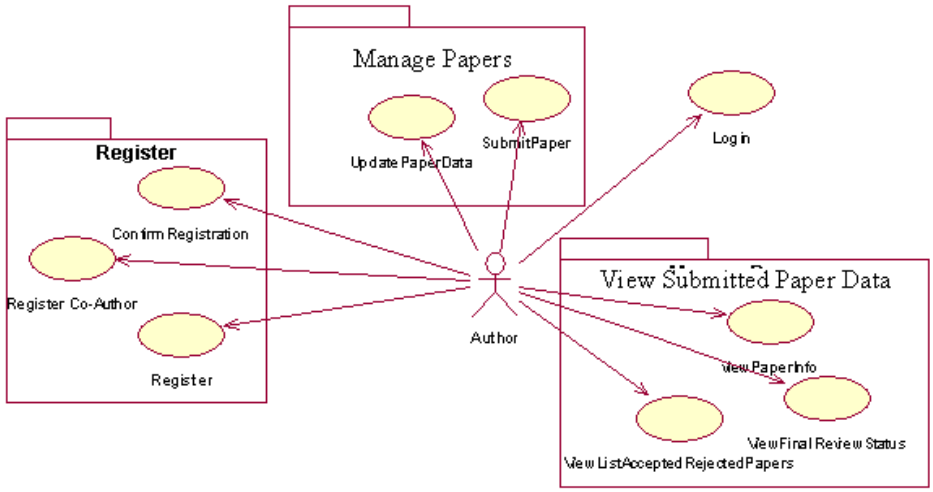

Fig. 27: Author's Use Cases grouping process

We have grouped the author related use cases into three groups: '*Register'*, that allows a given author to introduce the name of the people that have coauthored a paper, *'Manage Papers',* that provides any of the authors of a paper with the necessary functionality for them to update the data and/or the paper, and *'View submitted papers data',* that defines the reports the authors can visualize regarding the review process.

AUTHOR PROFILE ENTRY POINT

The NAD presented in Fig. 28 captures the functionality identified when the Use-Case grouping process was performed.

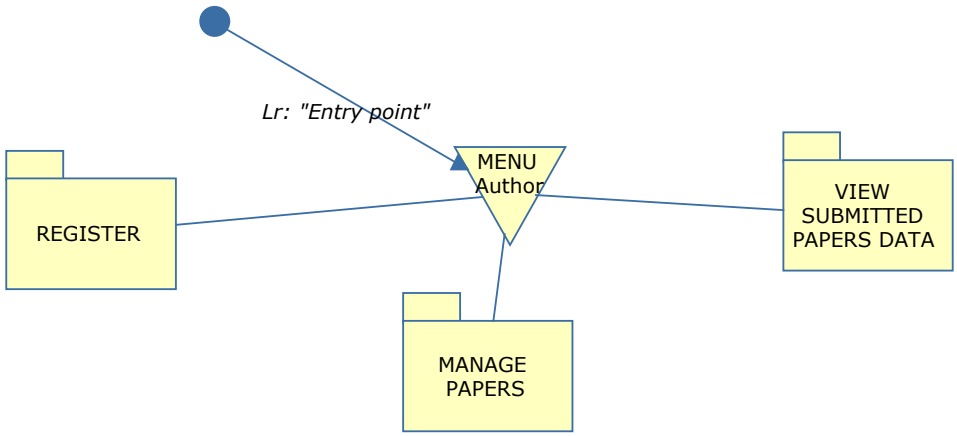

Fig. 28: Authors NAD. Level 0

NAVIGATION TARGET 'REGISTER'

An author can register himself or be registered by a coauthor of a paper, in which case s/he must confirm her registration. All three methods imply an identification of the author, that provides him with access to her profile functionality.

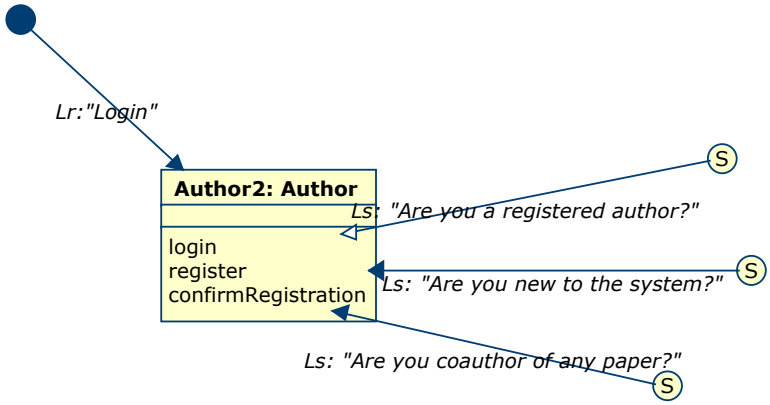

Fig. 29: Register

NAVIGATION TARGET 'MANAGE PAPERS'

This NT gathers track, subject and author method parameters by selection on the corresponding classes. An author can submit a new paper or modify any data about her already submitted ones. The constraint that allows an author to work only with her papers is implicitly preserved by the activation of the structural relationship from author (actor binded to the actual NAD) to paper. This relationship acts as a destination filter (Fd) for the links arriving at the paper class.

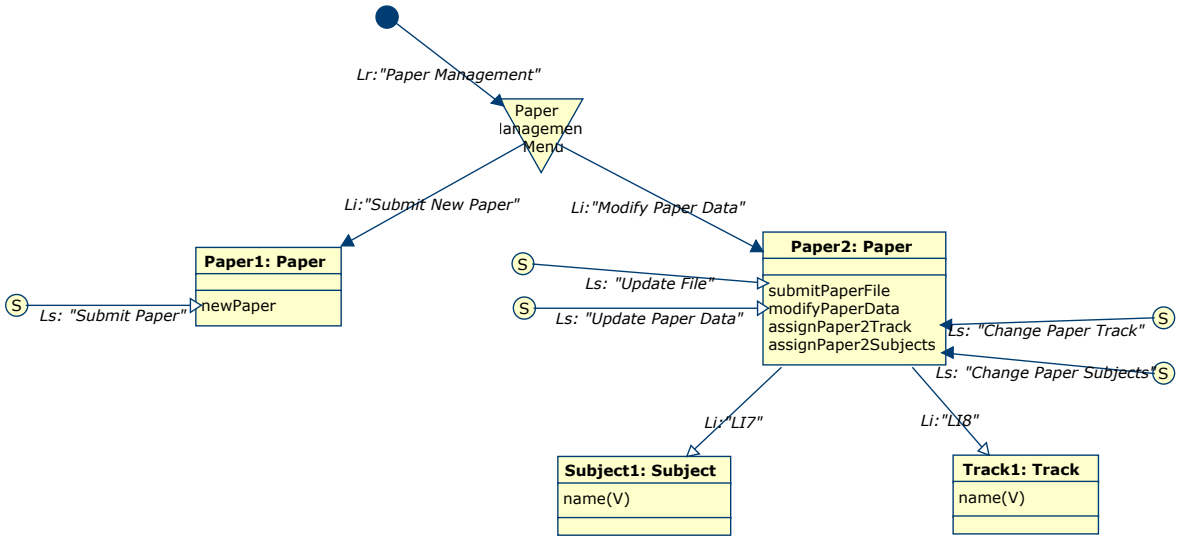

Fig. 30: Manage Papers

NAVIGATION TARGET ‘VIEW SUBMITTED PAPERS DATA’

Once the revision process has finished, the author can see the final revision results (see Fig. 31).

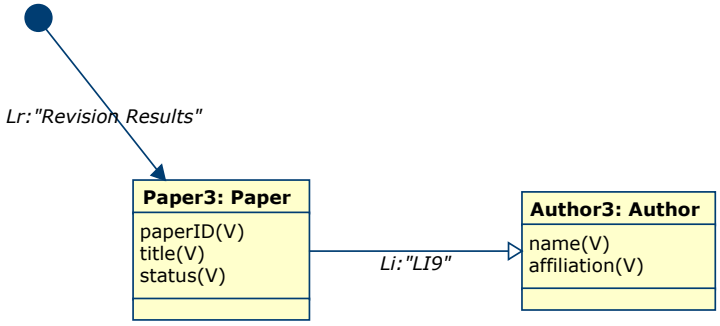

Fig. 31: View Submitted Papers Data

5.- View Refinement: the Abstract Presentation Diagram

5.1. Introduction

OO-H recognizes the need for a greater level of sophistication in interfaces than that provided by the NAD diagram, regarding both appearance and usability features. The Abstract Presentation Diagram (APD) provides a mechanism to refine the interface at a lower, more design-oriented level of abstraction. A default APD can be automatically derived from the NAD diagrams, and it reflects the abstract page structure of the interface defined at NAD level. Furthermore, it separates the different features that contribute to the final interface appearance and behaviour by using a page taxonomy, based on the concept of templates and expressed as XML documents, which are, namely: (1) Tstruct, used to capture the information that needs to be shown, (2) Tform, used when the page, apart from information, includes calls to underlying logic, (3)Tlink, that captures the interconnection and dependencies among pages, (4)Tfunction, that gathers client functionality used in the different pages, (5)TExternal, used to gather type, location and behaviour of external elements (such as images, applets etc) that may refine the initial interface, (6)Tlayout, where the location of elements and the definition of simultaneous views and its syncronization is captured and (7)Tstyle, where OO-H maintains features such as typography or colour palette for each element of the interface.

The default APD gives a functional but rather simple interface, which will probably need further refinements in order to become useful for its inclusion in the final application. It can however serve as a prototype on which to validate that the user requirements have been correctly captured. The refinement process consists thus on the modification of the default APD structure. This process is greatly simplified with the application of a series of APD-related patterns captured in the OO-H Catalog [4]. Furthermore, the catalog provides an executable python routine for each materialization of the patterns that OO-H calls 'Transformation Rule'. This routine, when loaded in the OO-H CASE tool and executed, changes the contents of the required APD abstract pages. Also, these routines can cause any new page, link or dependency to appear on the diagram. OO-H provides yet another way to manipulate some of the abstract pages (namely Texternal, Tlayout, Tstyle), by means of the Composite Layout Diagram (CLD). Despite its name, this view is not a diagram but a visualization of the interface prototype where the location and visual style of elements can be edited, and new elements can be added to improve the visual impact of the final generated interface.

The last step of the process, once the abstract pages (XML documents) have been refined, is to feed that device-independent modeled interface to a model compiler (not discussed here) that has the target-environment knowledge that allows it to generate an operational web interface.

In the next section we are showing part of the default APD generated from the PCChair NAD diagram.

5.2. APD Example: The PCChair Default Interface Siteview

The OO-H CASE tool includes an algorithm to generate, departing from the NAD diagram, the set of pages that make up the siteview of the interface. Note how only Tstruct,Tform and Tlink pages are relevant to the siteview of the system, and so how the APD diagram graphically shows them. OO-H includes a default Tfunction (client-side logic to control user inputs for method parameters), a default Tlocation and a default Tstyle.[8] Also note how the contents of the Tlink global page is shown by means of links connecting the Tstruct and Tform pages.

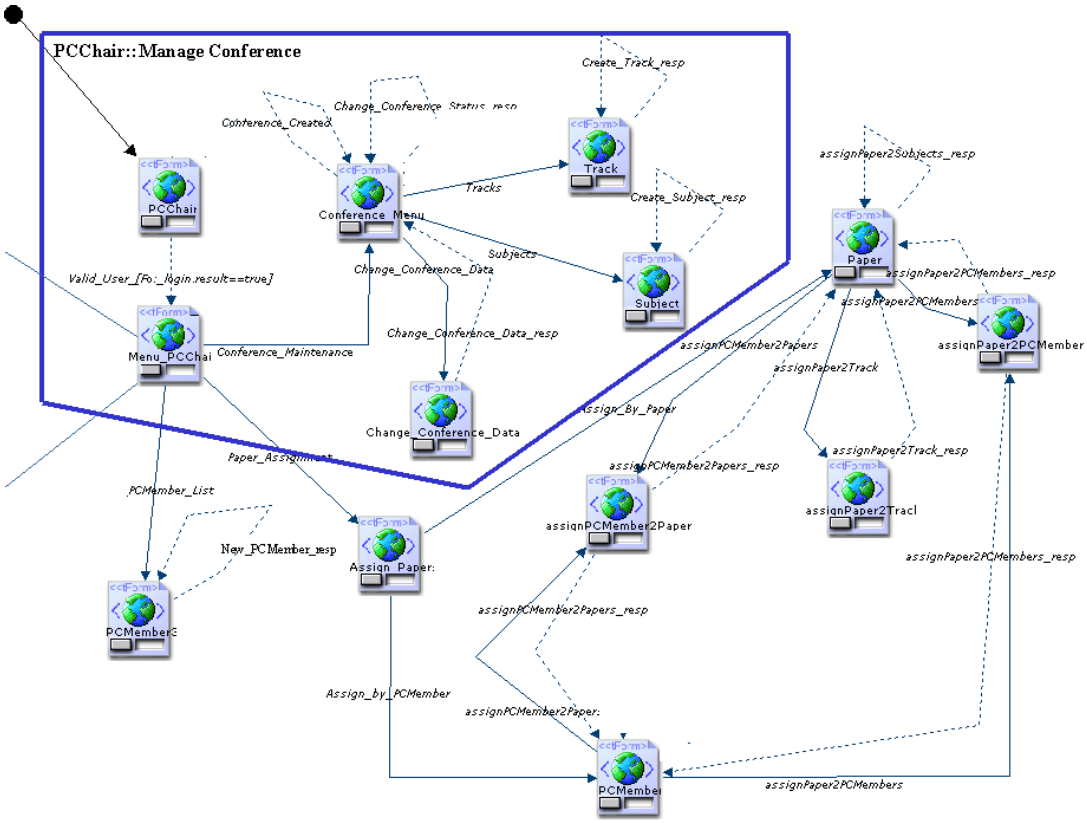

Fig. 32: Partial View of the APD corresponding to the PCChair Profile

If we look back to Fig. 9 (storyboard snapshot interconnection diagram) and compare it with Fig 32 (APD partial view), we can see how they match: the only difference is that snapshots S2 and S3 (Fig. 9) are now gathered in a single APD page ('*Conference_Menu*') due to the fact that, in Fig. 8 (NAD diagram), we defined that both views were shown in origin, and so embedded in the same abstract page, represented by the collection '*Conference Menu*'. The remaining snapshots can be directly associated to exactly one abstract page at the APD level.

[8] The XML specification of every page is available through the tool, either clicking on the corresponding page or by means of a menu

6.- Further Work

OO-H is not a closed proposal: the modeling of existing and to-come applications will cause its semantic constructs to evolve and/or be extended to capture new interface requirements. As an example, we are already working on the interface-related event modeling. OO-H distinguishes two types of event:

a. Logic events: events caused by the interaction of the interface with the underlying logic modules. Right now OO-H is able to model just synchronous services. However, we are interested in, if the underlying technology allows it, send a service request and continue working without waiting for the service to finish. This requirement implies the interface to listen to 'service-finished' events in order to warn the user and get results back. Also, we are interested in modeling events that are caused by facts external to the user, either punctual or periodic: in the Paper Review System one such example is that of deadlines that should cause the system to automatically change the state of the conference. As an example, we are interested in modeling the fact that each morning the system sends an email to every PCMember to remind him about the reviews s/he still has left, and how s/he is supposed to finish them. The system could also send a 'Call For Papers' reminder 15 days before the deadline for paper submission is reached.

b. Interface events: In the same way, we are interested in modelling which views are available for each user at each moment. One possibility is the use of high-level statecharts where each state represents a set of views available for a given user at a given time. A met transition condition implies a change in the set of views the user can access. Also, we are working on synchronization of views.

However, our main concern right now is the generation of mediators for the interconnection with pre-existent logic modules. In order to get this aim, OO-H defines the <<legacy>> stereotype that, associated to domain classes, reflects the fact that tose classes are part of a pre-existent library. When the generation process is executed, the designer must enter data about the physical location of the library, the protocol to connect to the library interface etc, that is, all the solution-space parameters that facilitate the generation of the mediator. It is thus important to make sure that the <<legacy>> classes capture the exact interface the library provides.

Other open ends in OO-H are (1) how to achieve a greater level of personalization and (2) detection of a greater range of patterns and definition of their corresponding OO-H Transformation Rules.

7.- References

1. eXtensible Markup Language (XML). http://www.w3.org/XML
2. E. Gamma, R. Helm, R. Johnson and J. Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.
3. J. Gómez, C. Cachero and O. Pastor. *Extending a Conceptual Modelling Approach to Web Application Design*. In CAiSE'00. 12th International Conference on Advanced Information Systems, volume 1789, pages 79-93. Springer-Verlag. Lecture Notes in Computer Science, 06 2000
4. J. Gómez, C. Cachero and O. Pastor. *Conceptual Modelling of Device-Independent Web Applications*. IEEEMultimedia. Special Issue on Web Engineering, pages 26-39. IEEE 04 2001.
5. OMG Unified Modelling Language Specification. http://www.rational.org/uml 06 1999
6. J. Warmer and A. Kleppe. *The Object Constraint Language. Precise Modelling with UML*. Addison-Wesley, 1998.