RESEARCH ARTICLE OPEN ACCESS

The EAOO-H Design Model for Transformation of WebGRL based Web Applications

Sangeeta Srivastava[1], Rohan Rajiv Saxena[2], Vandana Gupta[3]
[1]*(Department of Computer Science, BCAS, University of Delhi, India*
[2]*(Department of Computer Science and Engineering, IIT Kharagpur, India*
[3]*(Dept. of Computer Science, Kalindi College/ University of Delhi, India*

ABSTRACT
In case of systematic web application development. The first step is to capture the requirements comprehensively both explicit and implicit requirements. Further, in order to ensure a smooth conversion of comprehensive description of the requirements we need a design model that can handle the requirements captured earlier in totality. For this we need a design model that has the ability to present the different perspectives of a web application holistically. Therefore a new enhanced EAOO-H design model is presented in this paper that can capture both explicit and implicit requirements with the help of different web specific design models namely the content, navigation and the presentation models. These web specific models form a very important part of structured modeling of web applications. Therefore, we need to ensure that these web specific design models are such that they aid the conversion strategy from the requirements stage to the design stage and further convert easily into the construction stage. In this paper we present the EAOO-H model used for the transformation strategy for translation from the webgrl content model to the EAOO-H domain model.
***Keywords*:** EAOOH, Domain, Goal, Navigation, Presentation, WebGRL.

I. INTRODUCTION

Web applications form an essential part of our lives[1],[2] however, their development lacks a systematic development approach. The different phases of the Software Development Life Cycle like the requirements engineering and the design engineering have been either skipped or done in a hurried manner as such we get a web application which is not well structured as in the case of conventional information systems. As a result, we need a systematic process of web based applications development [3], [4]. The first step of a structured web application development is that the requirements need to be captured comprehensively. In the second step, the design model used for the web application development should be such that it provides a seamless transformation from requirements phase in to the design phase. These two steps form a very significant part of a web application development in order to get a good quality and high stakeholder satisfaction level.

Presently, the spotlight is on the web application development directly therefore the requirements phase is completely overlooked and the goals of the users are comprehended by the designer. This results in creating misunderstanding and confusion for the user, not the "real" user requirements.This causes both the development and implementation problems for designers and increases the initial project budget. A good requirements approach for the web application would give the designer the ability to take decisions from the very beginning of the development phase. These decisions could affect the design of the website for meeting the real goal requirements and preferences of user type [5]and [6].

In our EAOO-H design models we use the GOREWEB framework for Goal Oriented Requirements to model statics and the dynamics of requirements especially where behavior can change in time of web application development systems. "The use of goal driven requirements analysis helps in capturing stakeholders' goals and the requirement clarification and the conflicts between requirements can be evaluated and the best design option selected to suit the requirements [7], [8]. Also there are specific webgrl diagrams developed from the base webGRL diagram in the requirements phase which helps in giving a detailed picture from different perspectives related to the web applications in the design phase.

The existing design approaches lack the seamless transition of comprehensive requirements namely the implicit and explicit requirements in the requirements phase to the design phase and also due emphasis has not been given to the requirements phase for web applications, especially the non-functional requirements. The design models based on the input from the base WebGRL diagram in the requirements phase helps in giving a detailed picture from different perspectives related to the web applications in the design phase. We need a design

approach suitable for handling both simple and complex web applications. Even though a number of design approaches are there for web applications, however they lack the a structured web application development method and overlook both functional and nonfunctional requirements needed for a good web application development. As stated earlier the use of GOREWEB framework in the requirements phase solves these problems. The GOREWEB framework generates a set of specific webgrl diagrams that not only capture the all-inclusive requirements but also take care of the conflict resolutions and provide alternative solutions.

However, for the transformation of these requirements in totality and losslessly we need a design approach that has the ability to capture the requirements presented in the requirements engineering phase and is also suitable for lossless and seamless conversion of these requirements into the design phase. An existing A-OOH design approach of [9], [10] is chosen as a base design model and further enhanced to provide the ability for capturing all-inclusive requirements in the requirements phase and enable smooth transformation from the requirements stage to the design stage for web application development.

The A-OOH design approach has the ability of capturing the different facets of web applications however, it does not capture the nonfunctional requirements, as such the implicit requirements of the different stakeholders are overlooked. Therefore, the A-OOH design model needs to be enhanced to capture both functional and nonfunctional requirements. In order to capture these functional and non functional requirements, we need to enhance the domain , navigation and presentation design models of AOO-H into the Enhanced Adaptive Object Oriented Hypermedia (EAOO-H) design models. The EAOO-H model firstly needs to be enhanced to capture the different facets of web application development comprehensively and losslessly provided as an input from the webgrl diagrams. Secondly, the output of the EAOO-H design model has to be such that it provides ease of transformation from the design model into the construction phase.

The contribution of the EAOO-H design approach is that it follows a structured method of web application development as in the case of conventional software development. Secondly, the enhancement of the AOO-H approach results in a wider requirements capturing capability, as a result we can capture both goals and softgoals representing the functional and non-functional requirements of the different stakeholders of the web application under development. This capability was lacking in the A-OOH approach. Thirdly it provides three different perspectives of a web application development by using three specific EAOO-H design models namely the domain model, the navigation model and the presentation model. Finally the output of the design model is a UML profile that can easily be implemented in the construction stage with the help of an object oriented language.

In this paper we begin by giving the details of the Enhanced A-OOH approach for the domain model and discuss the extensions to the A-OOH approach required to meet the smooth transition of content Webgrl in Section 2. In section 3 we present the enhanced EAOO-H Navigation Model and later on in section 4 we present the EAOO-H Presentation Model and thereafter conclude the paper in section 5.

II. THE EAOO-H MODEL

The basics of the EAOO-H approach is based on the "A-OOH (Adaptive Object Oriented Hypermedia method)"of [9]. The EAOO-H design approach follows the same workflow as the A-OOH and the "OO-H modeling method" of [11]. The EAOO-H design approach output results in "UML-profiles" of [12] therefore all the models are "UML-compliant".

EAOO-H design model considers the following workflows:

1. Requirements:

In this stage the requirements for each type of user are gathered, including the non functional requirements. This phase is done using the GOREWEB framework. The webgrl and the webucm are used to capture the "functional and non-functional requirements" of the web application under development. This forms the "base Webgrl diagram". In the second phase in order to get a more holistic view of the requirements with respect to the web the base webgrl diagram has been extended into the "specific webgrl diagrams" namely the content, navigation, presentation and other webgrl diagrams. We use these diagrams as the input to our design phase. The basis of using these specific webgrl diagrams is their ability to capture both the "functional and non-functional requirements" comprehensively. While most of the existing approaches focus on the functional requirement only.

2. Analysis and Design:

In this stage all the activities related to the analysis and design of the software product are included:

a. Domain Analysis:

The user requirements captured by the content webgrl diagram are Goal, Softgoal, Task, and Resource and the intentional links are namely the decomposition links, contribution links, means

end links and dependency links, the relevant concepts for the application are gathered besides the designer knowledge of the domain.

b. Domain Design:

The content webgrl model of domain analysis has to be refined in consecutive iterations with new goals, softgoals etc. stated above into the final webgrl content diagram with the help of the GOREWEB framework. This refined content webgrl diagram is used in the model transformation strategy for smooth transition from the content webgrl diagram into the EAOO-H domain model explained below.

c. Navigation Design:

The domain information along with the navigation webgrl diagram is the main input for the design navigation activity, which is done using the transformation strategy for the transition of navigation webgrl to navigation model. The "navigation model is captured by means of multiple Navigation Access Diagrams as in [10] representing the different navigational views where the navigational paths are defined to fulfill both the different functional and the non-functional requirements" and the organization of that information in abstract pages.

d. Presentation Design:

Once the logic structure of the interface is defined, the presentation webgrl diagram is used as an input to the transformation process for transforming the Presentation diagrams that allows specifying the location, appearance and additional graphical components for showing the information and navigation of each of the abstract pages.

3. Implementation:

Implementation is the final workflow considered in the EAOO-H design model where the final application is generated.

4. Test:

The goal of this workflow is to verify that the implementation work is as intended. The Steps 1 and 2 are done using Web GRL diagrams and the domain analysis results in the Web Specific GRL diagrams. With the help of EAOO-H design model we map the refined requirements analysis models to their respective design models. The considered Web engineering approach EAOO-H is expressed as a UML compliant class diagram. This has led to the enhancement and development of our own UML profile as in [8].

2.1 Discussion on the Extension of the A-OOH approach to EAOO-H approach

The reasons and the steps taken to enhance the A-OOH approach into the EAOO-H approach are:-

- The A-OOH approach is requirement based whereas our work is goal oriented therefore in place of task we extend the goal as well as soft goals to the stereotypes defined in the A-OOH approach into navigation and presentation stereotypes.

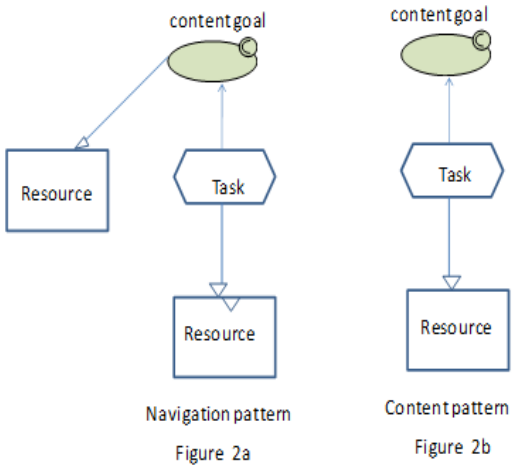

Figure.1: Showing Content Goals, tasks and Resources

- The A-OOH model uses the adaptive OOH approach to define the domain and the navigation model from the use case diagrams using domain analysis. We differ here by gathering the requirements and using gore approach to develop the specific WebGRL diagrams using webgrl approach for web applications.
- We do the requirements specification and analysis using the Webgrl approach and use EAOO-H only for the design phase to generate the domain model, navigation model and the presentation model based on the specific webgrl model .
- We extend the UML profile of the A-OOH approach by defining new stereotypes into the UML Profile for the EAOO-H approach to support these design models.

Once the requirements have been defined using the Web GRL diagram a transformation approach can be used to obtain the following design models for the website.

1."Domain Model (DM)":

The transformation strategy uses a set of rules to transform these web specific diagrams into the Domain model (DM), for defining the structure of the domain data.

2. "Navigation Model (NM)":

NM is used to represent the "structure and behavior of the navigation view over the domain data".

3."Presentation Model (PM)":

PM represents the "layout of the generated hypermedia presentation".

All of which are expressed using a UML compliant UML profile. In the next section the EAOO-H DM has been explained for the reader to easily follow the derivations of them.

III. THE EAOO-H "DOMAIN MODEL (DM)"

As the result of the domain analysis and domain design phases the domain model (DM) is defined. "It specifies the structure of the Web application domain data. The EAOO-H DM is a UML compliant class diagram. It give main points of the structure and functionality required of the relevant concepts of the application and reflects the static part of the system. The main models elements of a class diagram are the classes with their attributes, conditions and operations and their relationships"[13]. The EAOO-H domain model is also UML compliant class diagram. Though the main concepts are same as in A-OOH domain model however , it varies slightly in the description and the semantics of the domain model concepts. Now we present the domain model concepts and their description

3.1 The EAOO-H Domain Model Concepts

The EAOO-H domain model essentially consists of the basic modeling elements of a class diagram to capture the structure and the functionality of the Content Webgrl diagram and ensure a smooth transformation from the Webgrl content Model to the EAOO-H domain model.

The Domain Model is composed of the following concepts:-

3.1.1 Domain Classes (DC):

The Domain model consists of the domain class embodying the main objects, interactions in the application and the classes to be customized with an extensible program-code-format for object creation, giving preliminary values to state i.e part variables and executions of behavior using member functions or methods. The class stores the information content such as fields, attributes etc. The behavior of class is characterized utilizing methods. Operations are basically the subroutines for operating on objects or classes. These operations may lead to the alteration of the state of an object or basically give style of retrieving it. It is represented by a UML class with the stereotype << Domain Class>>. The class diagram is the main building block of domain modeling. The domain classes in a domain model consists of three sections shown in the Fig. 2 of domain class below.

3.2

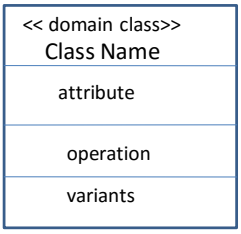

Figure 2: Domain Class

A class is made up of three parts as represented in the diagram above by a box with three segments:

a) The uppermost segment is used to represent the name of the class. The name of the class has a capitalized first letter and is printed in bold and centered.

b) The central segment represents the information content of the class and contains the attributes of the class. The attributes are written in lowercase with left-alignment.

c) The lowermost segment is used to represent the executable methods with the class and are written in lowercase with left-alignment.

An example of a domain class is shown below for the class name "Author". The author domain class is composed of attributes author name, author email and operation to display author list. A boolean condition can be imposed on the domain class for it to be valid as shown in fig. 3 below.

Figure 3: Example of the Author Domain Class

In the domain model design of the system, we pinpoint the classes and group them together for determining the static relations between those objects. As we proceed with the detailed modeling, the classes within the domain model are further divided into subclasses an example of the same is shown below. As shown in the fig. 4 below there are two domain classes namely author and category and the line shows the link or the association between them.

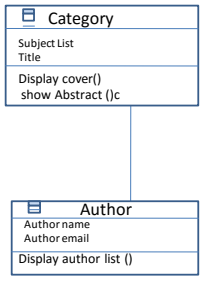

Figure 4: Example of relationships between domain classes.

3.1.2 Attributes:

defines "the structure of the class". "A class consists of information or data field explanations or properties, fields, data members, or attributes. These are usually field types and names that will be associated with state variables at program run time. These state variables either belong to the class or specific instances of the class" [14]. An example of the same is Author name: string.

3.1.3 Operations-

The "object class or its instance behavior is defined using operations. These are subroutines with the ability to operate on objects or classes. These operations may alter the state of an object or simply provide ways of accessing it. Many kinds of operations exist, but support for them varies across languages. Some types of methods are created and called by programmer code, while other special methods—such as constructors, destructors, and conversion operators—are created and called by compiler-generated code" [14]. An example of the same is the operation name +add to cart(int,float):void with input of the type int, float and initialization as void.

3.1.4 Relationship-

"Referencing between one or more related elements is represented using relationships. There is no general notation for a relationship. In most cases the notation is a line connecting related elements. Subclasses of relationship are association and directed relationship. A directed relationship is relationship between a collection of source elements and a collection of target elements. There is no general notation for a directed relationship. In most cases the notation is some kind of line drawn from the source to the target. Specific subclasses of the directed relationship define their own notation. Subclasses of the directed relationship are generalization and dependency. Generalization is a directed relationship between a superclass and a subclass. Dependency is a directed relationship which is used to demonstrate that some UML element or a set of elements requires, needs or depends on other model elements for specification or implementation. Dependency is a relationship between named elements as given in" [14]. "An association represents a family of links. A binary association is normally represented as a line. An association can link any number of classes to depict a relationship between them. An association can be named, and the ends of an association can be adorned with role names, ownership indicators, multiplicity, visibility and other properties. There are four different types of association: bi-directional, uni-directional, Aggregation and Reflexive. For instance, an author class is associated with a category class bi-directionally. Association represents the static relationship shared among the objects of two classes " [14]. Refer to figure 4 above.

3.1.5 Check:

"A constraint is a packageable element representing some check condition, restriction or assertion related to some element that owns the constraint or several elements". "Constraint is usually specified by a Boolean expression which must evaluate to a true or false. Constraint must be satisfied by a correct design of the system. Constraints are commonly used for various elements on class diagrams. In general there are many possible kinds of owners of a constraint. Owning element must have access to the constrained elements to verify constraint. The owner of the constraint will determine when the constraint is to be evaluated" [14]. For example, an operation can have pre-condition and/or a post-condition constraints. A Constraint is of the form '{' (name':') boolean expression '}' . Some of the languages used to define constraints are OCL, Java or any other constraint language .An example of the same is show below in fig. 5 and fig.6. For example a constraint on a class attribute can be *+author: String {author->notEmpty()}*

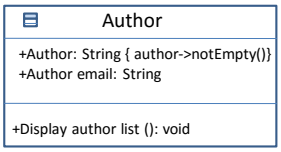

Figure 5: Constraint on Attribute

A constraint can be applied to an association or a class and is represented by a constraint string near the name or the symbol for the element. If a constraint applies to two elements, for example, two associations or to two classes then it is represented by a dashed line with an arrowhead between the elements with the constraint string labelling in curly braces. The direction of the arrow represents important information as the tail element of the arrow is mapped to the first position and the

head element is mapped to the second position in the constrained Elements collection.

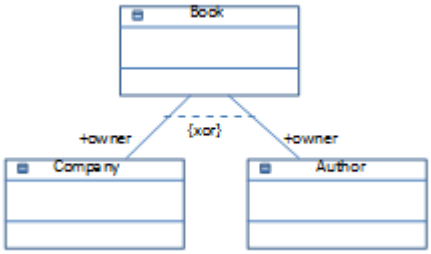

Figure 6: Constraint on association between classes

The different elements of a Domain Model have been described above. These provide a conceptual view of the web application, which is a very vital part of the web application development. The data or the content required within the web application is captured with the help of domain model concepts like Domain classes, relationships etc. Another important aspect of web application is the navigation within the website. This forms an intrinsic part of the web application development. The navigational views of the requirements captured by the navigational webgrl diagrams is represented with the help of the Navigation Design model described in the section below.

IV. THE ENHANCED A-OOH (EAOO-H) NAVIGATION MODEL

Once the domain model of the web site with the help of the WebGRL has been done we need to consider the Navigation Design of the software product. In this section, we present the Enhanced AOO-H navigation model. This navigation model is an enhanced version of the "A-OOH Navigation Model" [7]. The extensions and tuning of the A-OOH model has been done to ensure that the specific webgrl diagram is captured in totality and there is no loss of information while transformation.

The domain information is the main input for the design navigation activity, where the navigational paths are defined to fulfill the different functional requirements and the organization of that information in abstract pages. As in the case of "A-OOH Navigation Model the structure and behavior of the navigation view over the domain data is defined" [9] and [10] in the EAOO-H design model also.

4.1 The Navigation Met model

The main activity of the navigation model is to capture the navigation requirements specified by the stakeholder. In order to do so we need to capture where the navigational paths are defined to fulfill the different functional requirements and the organization of that information in abstract pages. We present the navigation metamodel which forms the basis for the EAOO-H Navigation model design in the fig. 7 below.

The Navigation metamodel is made up of two basic concepts:- the navigation node and the . navigation link. The navigation node is used to present the source and target destinations for a navigation path. The navigation path is represented with the help of the navigation link. A navigation model can have any number of navigation nodes and navigation links. The navigation node is further specialised into three types of nodes namely the navigation class, menu and access primitives. The navigation path can be between any three of these. i.e. a navigation path maybe between a navigation class and a menu or a navigation class and an access primitive or vice versa. All possible combinations of navigation between the three types of classes are permitted. However in case of a menu having multiple navigation paths to different navigation class, each navigation path between the menu and a navigation class will be represented separately by as many navigation links as the navigation paths from the menu to the different navigation classes.

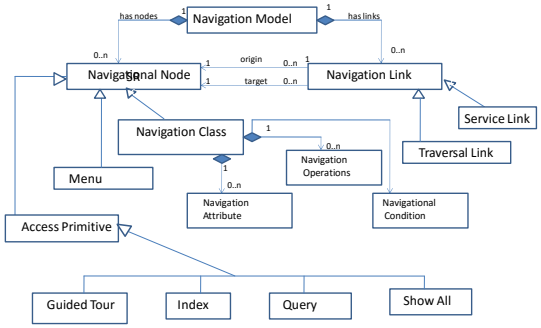

Figure 7: The Navigation Meta model

Each navigation class is composed of attributes and navigation operations. A navigation class may have one or more navigation operations. Navigation attributes are used to store the properties of the navigation class and navigation operations are used for display or to carry out a procedure for that navigation class."Access primitive can be an Index , a Guided Tour, Showall or Queries which collaborate in the fulfilment of every navigation requirement of the user . further there is a Menu or a Collection of hierarchical structures defined in Navigational Classes" [13]. The most common important of collection type is the concept of main grouping Navigational Links in various navigation course. The navigation tie-in is used to represent relationship between two navigation classes. Navigation links are of two types namely the traversal link and the service link. In case of a navigation link is used to navigate from the source to target navigation class for traversal or simple

process like display the target navigation class it is a traversal link. However if the navigation link results in an update of the logic while traversing from the source to the target navigation class in service link.

4.2 Navigation Access Diagram(NAD)

An EAOO-H Navigational Model (NM) describes a navigation view on data specified by the Domain Model. In AOO-H and OO-H the NM is captured by one or more Navigation Access Diagrams (i.e. NADs). The designer should construct as many NADs as different views of the system are needed, and provide at least one different NAD for each identified (static) user role. The AOO-H Navigation model the NAD is composed of Navigational Nodes, which represents a restricted view of the domain concepts, and their relationships indicating the navigation paths. The A-OOH Navigational model has been enhanced to transform the WebGRL navigation diagram. The EAOO-H Navigation model used by us 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) Access primitive which can be Index , Guided Tours, Showall or Queries which collaborate in the fulfillment of every navigation requirement of the user and (c) Menus or Collections (which are (possible) hierarchical structures defined in Navigational Classes. The most common collection type is 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 or the Means End Link (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. We further enhance the navigation links to represent the contribution link and decomposition links of the navigation Webgrl diagram. The contribution link of the navigation webgrl diagram will be represented as a traversal link or a service link depending on the contribution provided by the task and a decomposition link between a parent goal into sub goals is represented in the EAOO-H navigation model by a navigation target with service link which is represented by a link to an access primitive or simply by an access primitive alone . The various navigation model concepts used by us for the transformation of the navigation webgrl diagram into the EAOO-H navigation model are explained below.

4.2.1 The Navigation Node

Every navigation node is associated to an (owner) Root Concept from the DM attached to it by the notation: “Node:DM.RootConcept”. "Navigation nodes" are of three kinds: namely the Navigational Classes, Access primitive and Menus or Collections.

- ***Navigational Classes (NC):***
  These are domain classes consisting of attributes and operations whose the visibility is dependent on the permission for access to the class members.

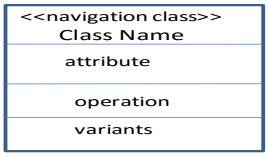

Figure 8: Navigation Class

It is denoted by a UML class embodied by a “stereotype the << Navigation Class>>” as shown in Fig. 8 above.

- **Access primitives-**
  Access primitives is required to access navigation objects to satisfy the navigational requirements of the user.So It is defined by the UML stereotypes:Index, Guided tour, Showall and Query. These stereotypes and icons stem from (19) and Isakowitz, Stohr and Balasubramanian (23) are defined below:

- **Index** –
  Index is a compound object, with one or more "index items" . Each "index item" is a named object, with a link to the "instance of a navigation class". "Each index is a member of some index class, which is stereotyped" by «index» with a corresponding icon. "An index class must be built to conform to the composition structure of classes" [12]. With the help of an index the instances of a navigation class are permitted to have a direct access as shown in fig. 9 below. In

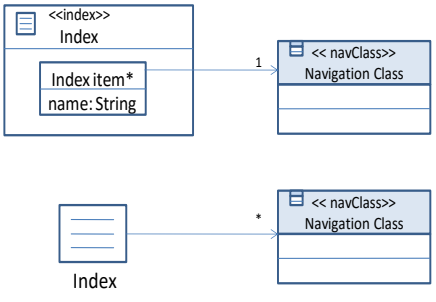

Figure 9: Index Class and shorthand for index

the short form the association between Index and Navigation Class is derived from the index composition and the association between IndexItem and NavigationClas as in [13].

- **Guided tour** –
  A guided tour is an ordered access to "instances of a navigation class". "For classes, which contain guided tour objects we use the stereotype" «guidedTour» and its corresponding picture depicted in Fig. 10. "Any guided tour class must be built to conform to the composition structure of classes" as in [13] shown in the Fig 10. below. Each Next Item must be connected to a "navigation class". "Guided tours may be controlled by the designer or by the system. Fig. 10 shows the shorthand notation for a guided tour class" .

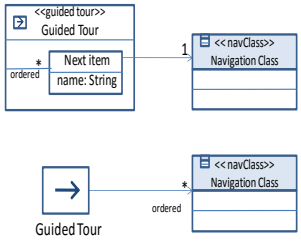

Figure 10: Guided Tour class and shorthand for guided tour

- Q**uery**- A "query is a class with query string as an attribute. This string may be given, for instance, by an OCL select mathematical operation"."For query classes we use the stereotype" «query» and the icon depicted in Figure 11 below.

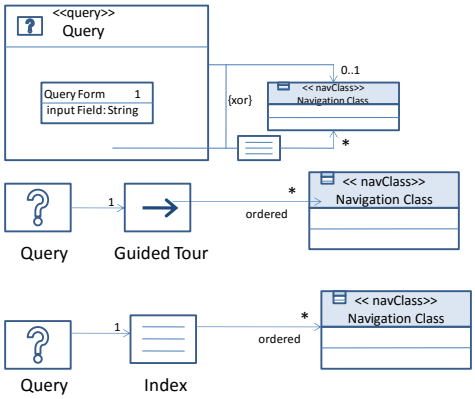

Figure 11: Query Class and Shorthand for Query

Any "query class" is part of "two directed associations related by the constraint {xor}". "Query with several result objects is modelled" to leading first to an index supporting the natural "selection of a particular instance of a navigation class". The "query" result can "alternatively be used as an input for a guided tour". Figure 11 above also shows the shorthand annotation for a query class in combination with an index class or with a guided tour.

- **Show All** -*:* A show all provides navigation without indexing and without internal navigation, all the objects are shown in the same abstract page. This is modeled by the stereotype<<Showall>> and its corresponding icon is depicted in the Fig. 12 below.

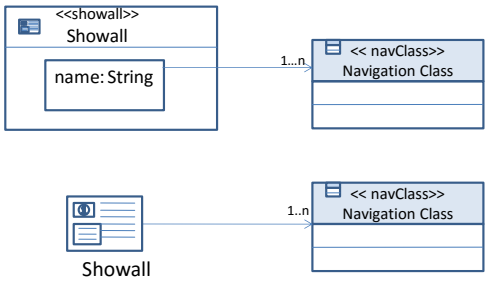

Figure 12: Showall Class and shorthand for showall

- **Menu:-** "A menu is a composite object" which contains a collection of navigation classes and navigation links represented by a "fixed number of menu items. Each menu item has a constant name and owns a link either to an instance of a navigational class or to an index, guided tour or query". Another most common collection type is the concept of menu grouping navigation links.

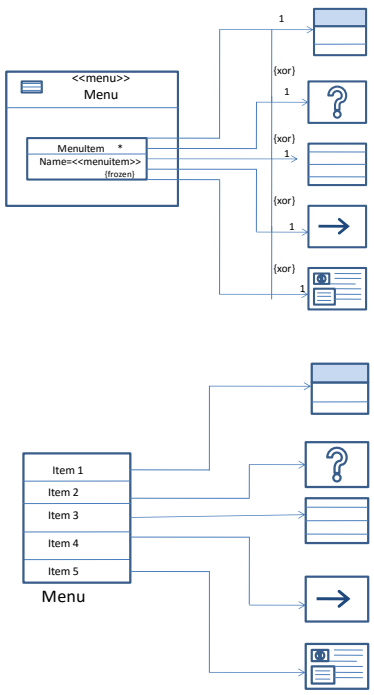

Figure 13: Menu class and shorthand for Menu

A menu class which is stereotyped by «menu» with a corresponding icon as shown fig. 13. A menu class should conform to the composition structure of classes described earlier.

4.2.2 Navigational Links (NL)

The Navigation Link *is* used to define the navigational route which user can follow through the system of rules. "A-OOH defines two main types of links:Transversal link : It is defined between two navigational nodes and Service Links for example navigational class, collection or access primitives as in [13] and Service or Means End Link : Navigation is performed to activate an operation which modifies the business logic and moreover implies the navigation to a node that displays data when the execution of the service is finished" [13]. We further enhance the navigation links to represent the contribution link and decomposition links which will be represented as associations in the navigation model .

- **T-Links (Transversal Links)-** "They are defined between two navigational nodes (navigational classes, collections or navigational targets). The navigation performed is done to show information through the user interface, without modifying the business logic. This type of links is represented by the stereotype <<TransversalLink>>" of [10] as shown in fig.14.

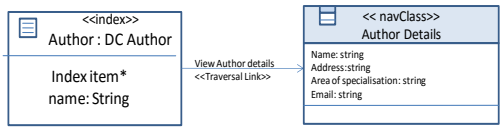

Figure 14: Transversal Link

- **S-Links (Service Links)-** "Navigation is 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. It is established when a service of the navigational class is activated. This type of links is represented by the stereotype <<ServiceLink>> and has associated the name of the invoked service" as in [10].

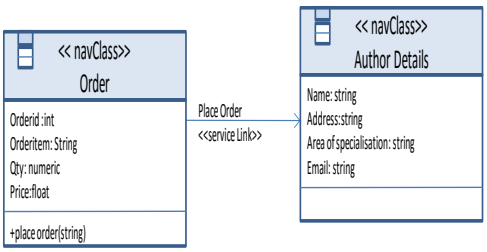

Figure 15: Service Link

In case the navigation performed is done to show information through the user interface, without modifying the business logic, then the navigation link is represented by a traversal link. Further, If a navigation link is also a decomposition link then as many traversal links between NCs are added as the number of decompositions of the navigation goal into its navigation sub goals represented by access primitives . However, if the navigation link is a contribution link then it is represented by a traversal or service link depending on the goal requirements.

The different elements of a Navigational Access Diagram(NAD) described above are used to form the NADs for the software product. These provide a navigational view of the web application, which is a very important part of the web application development. Different views result in more than one NADs. The greater the number of views presented with the help of the NADs more the exhaustive study of the different navigations required by the stakeholders for the software product.

V. THE EAOO-H PRESENTATION MODEL

In this part, we briefly present the EAOO-H Presentation model in order to ensure that the transformation of specific WebGRL models is flawless and all the elements of the WebGRL model are well represented in the enhanced AOO-H design models. Presentation Design: Once the logic structure of the interface is defined, EAOO-H allows specifying the location, appearance and additional graphical components for showing the information and navigation of each of the abstract pages. This is presented with the help of a Design Presentation Diagram consisting of two different levels.

We have already discussed the Content and the Navigation Webgrl models and their transformation to EAOO-H Domain and Navigation Design Models in Sections 3and 4. In this section we first present the EAOO-H Presentation Model and its concepts.

5.1 The Presentation Design Model

The Presentation model represents the layout of the generated hypermedia. During the presentation design, the concepts related with the abstract structure of the site and the specific details of presentations are gathered. The Presentation Model is defined in this activity. It can be captured by one or more "Design Presentation Diagrams (DPDs), for every NAD in the system there is a corresponding DPD". In the next section we present the Enhanced Design Presentation Diagram (EDPD) Metamodel and the description of its main elements.

5.2 The EDPD Metamodel

The DPD MOF metamodel has been defined to formalize the elements of the DPD and the existing associations among them as in [15] shown in fig. 16. A metamodel defines the language to express the model. There are two main basic elements of a DPD Metamodel namely the Presentation Nodes and the relationships among them represented by the Presentation Links.

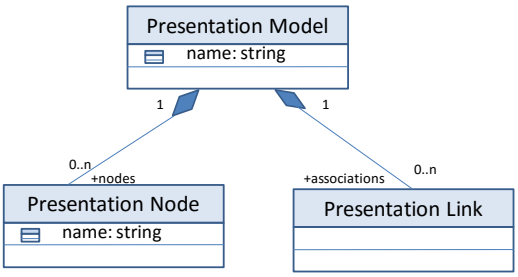

Figure 16: DPD Main Elements

There are two types of presentation nodes, the structure node and the layout node. The structure and the presentation nodes are considered at two different levels of the DPD. The structure nodes are used to define the structure of the presentation of the website and are considered at level zero of the DPD. The Layout of the presentation model for the web application is considered at level one. The different types of structure and layout nodes are described briefly below.

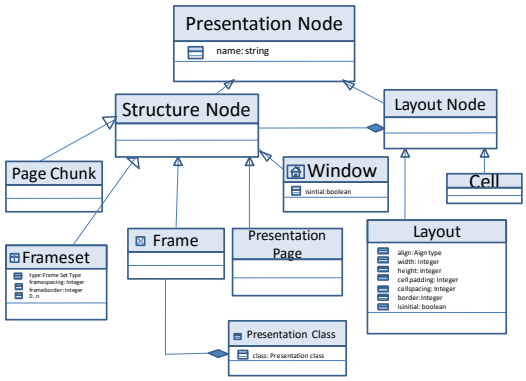

Figure 17: Presentation Node Subtypes

The different types of Structure Nodes are: Presentation Page, Window, Frame, Frameset, Presentation Class and Page Chunk (level zero of the DPD) .Similarly the different types of Layout Nodes considered are: Layout and Cell (level 1 of the DPD) as shown in fig. 17.

Just like the presentation nodes there are a variety of presentation links to represent different types of relationships between the presentation nodes. In a Presentation Model five types of Presentation Links can be defined as shown in fig.18.

- Navigates: This relationship can be defined between Presentation Page elements.
- Builds- This relationship is defined between server page and client page where a server script is used to build the client page.
- Submit :It is defined as the relationship between the form and the server page where the input content of the form is submitted to the server page.
- Contains: This relationship is defined between Presentation Page elements and Page Chunk elements to indicate a presentation chunk is contained (and shown) in one or several Presentation pages.
- Redirects:- This relationship can be defined for a Presentation Page that redirects to itself.
- Displays or Presents:-This relationship can be defined between a Presentation Page and its frame elements or a window and frame on which the presentation page is displayed.
- IncludeFrame: This relationship can be defined between FrameSet and Frame elements.
- IncludeCell; This relationship is defined between Layout and Cell elements

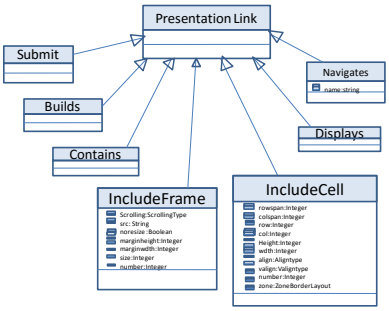

Figure 18: Presentation Link types

The Enhanced DPD (EDPD) Metamodel and its elements have been described in this section in brief we now present the elements of the EDPD in detail in the next section.

5.3 The Enhanced Design Presentation Diagram

The basis of the enhanced DPD described below is the A-OOH DPD. However as per the requirements of the WebGRL diagram and to ensure faultless transformation from WebGRL diagrams to EAOO-H based models we have enhanced the existing A-OOH DPD to the Enhanced Design Presentation diagram (EDPD). The enhanced version EDPD and its modelling elements are described below.

There are two main goals of the EDPD, the first goal is to describe the organisation of pages within the website where navigation nodes of the NAD are assembled into the presentation pages. Presentation Pages are abstract pages which can be represented by converting them into one or more

concrete pages. The designer can use additional static pages to the EDPD also. This represents the level 0 of the EDPD.

The second goal is used to represent the layout and style of each page of the interface. The layout and style are used by the designer to describe the components, their style and their positioning on the page. The designer can later on change the individual page structure and add static elements to the EDPD. This shape the level 1 of the EDPD formed by expanding the abstract pages defined in the level 0 earlier. After the refinement of the EDPD the front end for the web application, static or dynamic can be generated depending on the constraints of the target platform. The main modelling elements of the EDPD are described next. We classify them into level zero or level one depending on where they can be defined.

5.3.1 Main Elements in the Level Zero of the EDPD

As aforementioned this level provides the assembling of navigational nodes into presentation pages that form the page structure of the website. Moreover new static paper can be added to this level. We enhance the existing DPD of A-OOH to represent the following:-

- **Presentation page**

This element corresponds to an abstract page which can be a Server or a Client Page with an associated Presentation model where we define all the components shown in that page. It is represented as a UML Package with the stereotype <<Presentation Page>> as in fig.19. Inside this package the Presentation Model attached to the Presentation Page is shown.

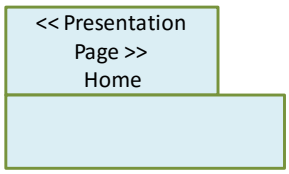

Figure 19: Presentation Page

- **Server Page** is used to specify content generated using a server script.
- **Client Page** is used to specify content generated using a client script which is presented or displayed in a target window. The position of the content on the presentation page is given by the window, frame or framesets.
- **Target** is an abstract class used to generalize the concept of window and frame on which the presentation page is displayed.
- **Window** is the part of the user interface where presentational interface components are displayed. The notation is a UML class with the stereotype <<Window>> as we can see in Fig.20 below.

All the elements described above are presented in the fig 20 below.

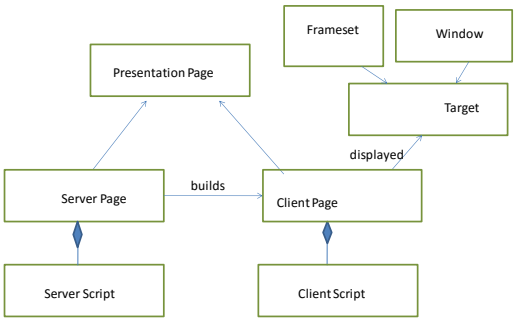

Figure 20: Main Elements in the Level 0 of the EDPD

- **Page Chunk**

The Page Chunk element defines a section of an abstract page. The constituents of the pagechunk are defined in the associated Presentation model. This section can be used again in other pages of the Web application as such repetition is avoided. It is symbolised by a UML Package with the stereotype <<Page Chunk>> in fig. 21. The Presentation Model attached to the Page Chunk is shown inside it.

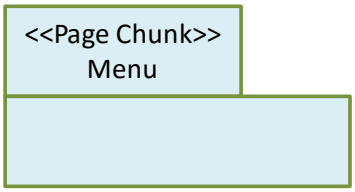

Figure 21: Page Chunk

- **FrameSet, Frame**

The FrameSet element represents a set of frames in which the browser window can be divided. In this way we provide the designer the possibility of using a frame based design for his application. It is symbolised by a UML class with the stereotype <<FrameSet>> as seen in fig 22. The Frame element represents a frame that is part of a FrameSet. It is symbolised by a UML class with the stereotype <<Frame>>. FrameSet and Frame elements are associated with the <<includeFrame>> association as in [15].

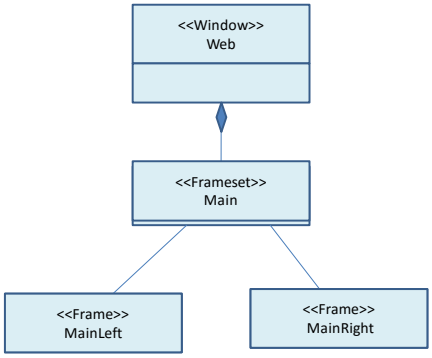

Figure 22: Window, FrameSet and Frame elements

5.3.2 Main Elements In the Level One of the EDPD

As already explained, the goal of level one is to describe the layout and style of each page of the interface defined in the level 0. Static elements can be added to the pages like static text. If we use frames and framesets to present the content on a presentation page then each frame or a frameset has a presentation class associated with it.

- **Presentation Class**

The Presentation class is a structural unit which contains the user Interface Components shown in fig. 23 which are to be represented in that frame or frameset.

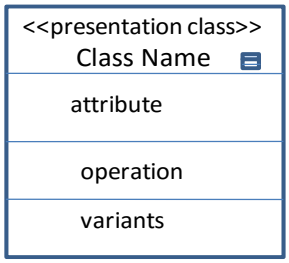

Figure 23: Presentation Class

- **Interface Components are text image video audio anchor and forms.**

❖ *Composed Interface Component*

The composed interface components considered are Collection, Anchored Collection and anchor

- ➢ Anchor- Anchor is a composed interface component which contains a simple interface component.
- ➢ Collection- Collection is a collection of simple interface components image, text, video and audio.
- ➢ Anchored Collection-Anchored Collection is a collection of anchors.

❖ *Simple Interface Component*

The simple interface components considered are:

- ➢ Image, Text, Audio, Video and Form Elements (e.g. Input Button, Input Submit…etc).

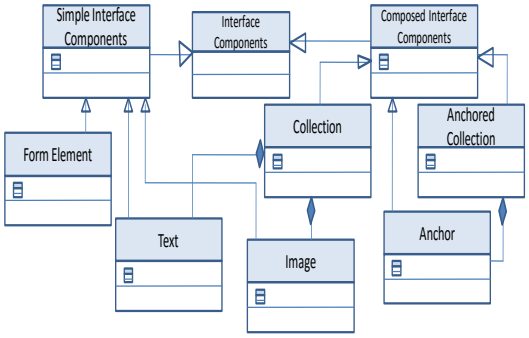

Figure 24: Presentation Class and the Interface Components

- **Layout**

Instead of using frames, layouts can be defined for defining the disposition of the objects visualized in the Web page. These layouts can be of three different types: BoxLayout, BorderLayout and GridBagLayout; each of them provides a concrete distribution for its cells. The layouts are translated, in last instance, to HTML tables. The different types of layouts considered can be nested.

❖ **Box Layout**

Box Layout arranges components on top of each other or row of your choice a typical box layout is shown in fig. 25.

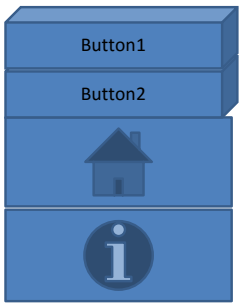

Figure25: BoxLayout

❖ **BorderLayout**

As the Fig. 26 shows, a BorderLayout has five areas in which we can place the different elements of a Web page.

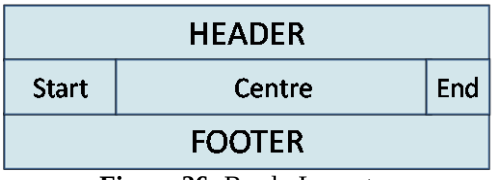

Figure 26: BorderLayout

❖ **GridBagLayout**

GridBagLayout is one of the most flexible layout. A GridBagLayout places components in a grid of rows and columns, allowing specified components to span multiple rows or columns. Not

all rows necessarily have the same height. Similarly, not all columns necessarily have the same width. Essentially, GridBagLayout places components in rectangles (cells) in a grid, and then uses the components' preferred sizes to determine how big the cells should be. The following fig. 27 shows an example of a gridbaglayout. As you can see, the grid has three rows and three columns. The button in the second row covers all the columns, whereas the third row button covers only the two right columns.

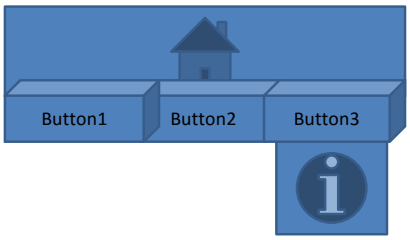

Figure 27: GridBagLayout

- **Cell**

A cell represents a component which is part of a layout. The cells act as containers for the interface components.

The two levels of the EAOO-H Presentation model have been described in the section above. With the help of the DPD we can represent the presentation goals of the presentation webgrl diagram of the requirements engineering stage in the design stage in totality.

VI. CONCLUSION

In this paper we have tried to present the EAOO-H design model for web application. The three specific design models namely the domain model, navigation model and the presentation model represent the three different facets of a web application. These design models are used to capture the different perspectives of the web application in a manner such that the transition from the requirements stage to the design stage is lossless and also the output of the design model is a UML Compliant, UML Profile which can easily be used for implementation of the web application using an object oriented language.

The advantage of the EAOO-H approach is that it captures the different facets specific to web applications in a comprehensive manner. Also it makes use of a structured method of web application development by moving through different stages of Software Development Life Cycle in a systematic way. Further the requirements input used for the EAOO-H design model being a webgrl diagram. We use a requirements engineering approach that captures the "functional and nonfunctional requirements" in entirety. The different webgrl diagrams provide the different perspective of the web application. The EAOO-H model has been enhanced and modified in a manner such that it has the capability of capturing both goals and softgoals for a web application. Also the output as stated above of these design models is a UML Compliant UML Profile that eases the transformation from the design stage to the construction stage and reduces the workload of the coding engineer.

In future we plan to define a transformation strategy from the requirements to the design stage. the development of the transformation strategy would lead to direct conversion of the requirements to design phase in an automatic manner. The entire process from the requirements to the design phase is done seamlessly and losslessly resulting in a lot of reduced workload for the design engineer, thereby generating a high quality product with lesser resources.

References

[1]. Cachero, C. and Gomez, J. (2002). Advanced conceptual modeling of web applications: Embedding operation interfaces in navigation design. JISBD, pp.235–248.

[2]. Ceri, S., Daniel, F., Matera, M. and Facca, F. (2007). Model-driven development of context-aware Web applications. ACM Trans. Inter. Tech., 7(1), pp.30-63.

[3]. Koch, N. and Wirsing, M. (2001). Software engineering for adaptive hypermedia applications. PhD. Thesis, ReiheSoftwaretechnik, pp.145-289.

[4]. Kraus, A. (2007). Model driven software engineering for web applications. München: Ludwig-MaximiliansUniversität, pp.73-114.

[5]. Chawla, S., Srivastava, S. and Bedi, P. (2011). GOREWEB Framework for Goal Oriented Requirements Engineering of Web Applications. In: IC3, CCIS 168. Berlin: Springer-Verlag, S. Aluru et al, pp.229–241.

[6]. Chawla, S., Srivastava, S. and Bedi, P. (2015). Goal and Scenario based Web Requirements Engineering. in International Journal of Computer Systems Science & Engineering.

[7]. Chawla, S., Srivastava, S. and Bedi, P. (2015). Improving the quality of web applications with web specific goal driven requirements engineering. International Journal of System Assurance Engineering and Management.

[8]. Chawla, S. and Srivastava, S. (2010). Goal oriented Requirements Analysis for Web Applications. In: International Conference on Computer and Software Modeling. Manila: IEEE, pp.88-92.

[9]. Aguilar, J.A., Garrig´os I, Maz´on, J. N., Trujillo, J. (2010). An MDA Approach for Goal-oriented Requirement Analysis in Web Engineering. Journal of Universal Computer Science, vol. 16, no. 17 (2010), 247, pp 2475-2494

[10]. Aguilar, J.A., Garrig´os I, Maz´on, J. N., A Goal-Oriented Approach for Optimizing Non-functional Requirements in Web Applications. (2011), Advances in Conceptual Modeling. Recent Developments and New Directions,Volume 6999 of the series Lecture Notes in Computer Science pp 14-23.

[11]. Gómez, J., Cachero, C. and Pastor, O. (2000). Extending a Conceptual Modelling Approach to Web Application Design. In: Proceedings of the 12th International Conference on Advanced Information Systems Engineering. London, UK: Ed. Springer-Verlag, pp.79-93.

[12]. Srivastava, S. (2015). UML Profile for the WebGRL Requirements Model and EAOOH Design Models. International Journal of Emerging Technology and Advanced Engineering, 5(8), pp.313-322.

[13]. Srivastava, S. (2014). Model Transformation Approach for a Goal Oriented Requirements Engineering based WebGRL to Design Models. International Journal of Soft Computing and Engineering (IJSCE), 3(6), pp.66-75.

[14]. www.uml.org (Accessed June 2016)

[15]. Srivastava, S. (2014). A Systematic Approach towards Transformation of Presentation Web Goal Oriented Requirements Language to Presentation Design. International Journal of Scientific and Engineering Research, 5(9), pp.7-17.