UML Profile for the WebGRL Requirements Model and EA-OOH Design Models

Sangeeta Srivastava

BCAS, Delhi University, New Delhi, India

Abstract—The growing need for web applications has resulted in their development in an adhoc manner. A systematic method of Information Systems development is not being followed. As a result the applications thus developed focus mostly on the design and presentation aspects and do not meet the real goal and expectations of the stakeholders. We present a Web based Goal Oriented Requirement Engineering based approach WebGRL for transformation of web specific models into their UML Profiles. Goal driven requirements analysis helps in capturing stakeholders’ goals very finely, they enhance the requirements analysis in many ways, as the requirement clarification and the conflicts between requirements can be detected at an early stage and design alternatives can be evaluated and selected to suit the requirements. In this paper, we take a step from the requirements phase to the design phase. While adhering to the web based goal oriented requirements engineering we move to the design phase using the intermediary EA-OOH design models supported by a UML profile. This helps in seamlessly generating the web specific UML profiles for the content, navigation and presentation Web GRL models. As the WebGRL approach uses repeatedly refined and resolved alternatives presented by us in the GOREWEB framework as an input to the UML profile this would lead to a better design and high quality of product development which captures the stakeholders’ goals very closely.

Keywords—Goal Oriented Requirements Engineering, UML Profile, Web Engineering, WebGRL, UML Profile for Content WebGRL Diagrams, UML Profile for Navigation WebGRL Diagrams, UML Profile for Presentation WebGRL Diagrams.

I. INTRODUCTION

In this age of internet, our increasing reliance on the net has resulted in titling the Information Systems development greatly in favour of Web Applications. The result is that web application development has mushroomed favourably. The web applications being developed today are varied, large and complex. They involve multiple stakeholders, and the size and purpose of the applications are also varied [1]. Web applications being developed currently involve important business strategies that are critical to their business systems. In order to ensure that these web applications are developed in a systematic manner we need to lay emphasis on Requirements Engineering. As such, we need a paradigm shift in the area of web application from the presentation angle to the other Information Systems development phases.

The traditional Information systems lay emphasis on the Requirements Engineering phase of the software development. Therefore, in the case of web applications too the focus must change from the presentation angle towards the requirements engineering phase.

A number of requirements engineering approaches exist, we have chosen the goal and scenario based requirements analysis approach namely the Goal Oriented Requirements Engineering (GORE) as it is a popular approach that captures both the functional and non functional requirements with the help of goals and softgoals respectively. Many approaches have been developed for Goal oriented Requirements Engineering for generic systems [2], [3], [4]. However, the notations and models developed for generic applications focus on the functional requirements as such they do not capture the non functional requirements adequately. Also they do not address some very important issues of web applications like content, navigation, presentation etc. Some work has been done by researchers [5], [6], [7],[8] on web engineering approaches taking into account the Goal driven analysis, but many concepts of goal driven analysis like design rationale, conflict resolution, goal prioritization have been overlooked therefore a holistic view is needed to incorporate these. For enhancing the requirements engineering activities involved in web application development, GOREWEB: Goal Oriented Requirements Engineering for Web applications framework has been used for goal oriented requirement analysis of web applications [9].

GOREWEB model extends the concepts of User Requirements Notation (URN) for comprehensive study of web application requirements. URN [10], [11] refers to User Requirements Notation. It is currently the only standard that combines goals and scenarios in one notation. It is a combination of two notations GRL (Goal Requirements Language) and UCM (Use Case Maps). User Requirements notation aims to capture goals and decision rationale that finally shape GOREWEB Framework for Goal Oriented Requirements Engineering models dynamic systems where behaviour may change at run time. GRL is Goal Requirements Language that focuses on Goal analysis. It help in defining the goals including the non-functional requirements, evaluating them, resolving conflicts etc. UCM stands for Use Case Maps that are the visual notation for scenarios. UCM notation employs scenario paths to illustrate causal relationships among responsibilities.
Numerous languages and notations have been developed over the years to model goals and their relationships in an explicit way. While some metamodels for goal modeling languages exist, it has often been completed in isolation and has not been done in accordance with the standards. Yet, there exists one mature metamodel for goal modeling, namely that of the Goal-oriented Requirement Language (GRL) mentioned above. More recently, the Goal-oriented Requirement Language (GRL) has become an internationally recognized standard for goal-oriented modeling, as part of a new Recommendation of the International telecommunications Union named User Requirements Notation (URN) [1], recently standardized as ITU-T Recommendation Z.151 [12]. URN being the latest and standardized notation for Goal and Scenario based requirements analysis, our work is based on this notation. An extended form of GOREWEB framework called WebGRE framework is being used as the basis for this work [9]. We use the refined WebGRL diagram of the WebGRE approach as an input to transform seamlessly from the requirements engineering phase to the design phase. In this paper we present the UML Profile for modelling GORE based WebGRL components into design model components. The advantage of using the WebGRE approach is that most of the problems have already been tackled in the requirements engineering detailed analysis phase and we have a very refined input to our transformation approach.

The Unified Modeling Language (UML) [13] is the standard prevalent modeling language for developing software applications due to its platform independence capability. The Unified Modeling Language (UML) does not fully address the needs of some important modeling domains, including goals and non-functional requirements (NFR). As a result of the increasing usage of UML in the industry, we need to address this compelling issue and enhance UML to describe and structure goals and NFR. UML can be extended and adapted through the definition of profiles. In this paper, we propose a UML profile for the Web Goal-oriented Requirement Language (WebGRL), which helps in mapping the specific WebGRL diagrams like the Content Webgrl, Navigation Webgrl, and Presentation WebGRL diagrams expressed using elements described in our work [9] through intermediary EA-OOH design models into its respective UML Profile. The transition of the WebGRL models into a UML compliant UML Profile aids in platform independent development of the product. The main benefit of our point of view is that the designer will be able to make decisions from the very beginning of the development phase. These decisions could affect the structure of the envisioned website in order to satisfy needs, goals, interests and preferences of each user or user type.

II. RELATED WORK

The related research work in the area of web design are many, like the Object-Oriented Hypermedia Design Method (OOHDM) [14], [15] and its successor, the Semantic Hypermedia Design Method [16], [17] that permit the concise specification and implementation of web applications. This is achieved based on various models describing information, navigation and interface aspects of these applications, and the mapping of these models into running applications, in various environments. Another work close to our approach is the UWE approach [18], [19] which is an object oriented approach that has as a distinguishing feature its Unified Modeling Language [20] compliance since UWE is defined in the form of a UML profile and an extension of the UML meta-model. The fundamentals of this approach are a standard notation (UML through all the models), the precise definition of the method and the specification of constraints (with the OCL language) to increase the precision of the models. All these approaches listed above are either not suited for the web applications as in OOHD and SHDM and if they do then they treat the web application development process differently. Besides, this they do not emphasize on the requirements part. The greater emphasis in all these cases is on the design phase of web applications. However, the web design approach used by us based on the WebGRE Framework for Goal Oriented User Requirements makes use of goal driven requirements analysis that helps in capturing stakeholders’ goals very finely, they enhance the requirements analysis in many ways, as the requirement clarification and the conflicts between requirements can be detected early and design alternatives can be evaluated and selected to suit the requirements [9], [21].

We present in this paper a UML profile for transformation from the specific Web GRL diagrams to their specific UML Profiles for the content, navigation and presentation WebGRL diagrams using the EA-OOH design model as in intermediary. We briefly present the related work in section 2. As a part of the WebGRE framework we have enhanced the GRL Metamodel to the WebGRL Metamodel of [8] presented concisely in section 3. Thereafter we present the EA-OOH Design models as intermediary for transforming them into their respective UML Profiles in section 4. In section 5 we present the UML Profile for the content model. In Section 6 we present the UML Profile for the navigation model. Further, in section 7 we present Profile for the presentation model. In section 8 a case study of transformation of an Online bookstore presentation WebGRL diagram to the enhanced EA-OOH Design Presentation diagram. In the last section we present the conclusion and future work.
III. WEBGRL METAMODEL

The GRL Metamodel has been enhanced for representing web specific functional and non-functional requirements through a goal driven approach, we have presented the Web GRL Meta model shown in the figure1 below in [9]. Also the Web specific User Requirements Notation which has been enhanced from URN is given in detail in [9]. WebGRL metamodel consists of Intentional Elements and Links. The intentional elements are Goal, Softgoal, Task, and Resource. The Goals and softgoals have been enhanced from GRL notation to suit the web specific needs. The notation has been enhanced to incorporate Web specific functional and non-functional requirement. The tasks & resources are represented in a similar way as in GRL. WebGRL notation also consists of links that connect two or more intentional elements. They are decomposition links, contribution links, dependency links and means end links.

After the BaseWebGRL diagram has been generated, for detailed analysis, it is refined for each functional requirement category of NFR i.e. for Content, Navigation, Presentation, Business Process and Adaptivity requirements. This results in web specific GRL diagrams. Thereafter using our transformation approach we transform these web requirements expressed using the WebGRL diagram into the web design phase using the enhanced A-OOH (EA-OOH) method of [22], [23] to generate the five models:-

- Domain Model
- Navigation Model
- Presentation Model
- Adaptation Model
- Business Process Model

The details of the refined WebGRL diagrams and the enhanced AOOH model is given in [8] and [22], [23] respectively.

The existing standard ‘User Requirements Notation’ (URN) has been enhanced to suit the web specific needs. The two parts of URN- ‘Goal Requirements Language’ (GRL) & ‘Use Case Maps’ (UCM) have been enhanced suitably. The extended notation for analysing goals is named as WebGRL and the Use case maps have been enhanced for incorporating navigation in the scenarios and termed as WebUCM. The enhanced notation clearly depicts Web specific Functional & Non-Functional Requirements has been presented in detail in [9].

IV. TRANSFORMATION TO EA-OOH DESIGN MODELS

The requirements analysis of the WebGRL results in the Web Specific GRL. A number of UML Profiles or UML compliant approaches exist for transformation from requirement phase to design phase like UWE, OOH, and OOHD M for web engineering. However, A-OOH modeling method is very close to our approach With the help of A-OOH we map the refined analysis models to their respective design models This approach is very close to the web specific diagrams generated in the analysis phase in our previous work so we have adopted this approach in this paper and enhanced it as well as modified it to EA-OOH approach of [22], [23] to define the web specific models of WebGRL in the design phase into their respective design models with traceability. This has lead to enhancement and development of our own UML profile. 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 softgoals to the stereotypes defined in the A-OOH approach into navigation, presentation, adaptation, and business process stereotypes. 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 WebGRL diagrams using GRL approach for web applications, we do the requirements specification and analysis and use A-OOH only for the design phase to generate the domain model, navigation model and the presentation model. In future we will enhance it to represent the business process and adaptation models and a UML profile to support these.

V. UML PROFILE FOR WEBGRL DIAGRAMS

In this section a brief introduction to the extension mechanism of UML is explained along with the UML Profile for the WebGRL diagram in the figure 2 below.
The UML Profile of the WebGRL diagram is based on the definitions and the diagrams explained in [30]. Profile Diagram is a structure diagram which describes lightweight extension mechanism to the UML by defining custom stereotypes, tagged values, and constraints in [24],[25],[13]. Stereotypes, tagged values and constraints represent the built-in extension mechanisms of UML. UML represents a family of modeling languages, rather than a single language. By Lightweight we mean that it can be easily supported by tools and it does not impact the interchange of formats as stated above. Profiles allow adaptation of the UML metamodel for different platforms.

Semantics of a standard UML metamodel elements could be specialized in a profile. Profile only allows adaptation or customization of an existing metamodel with constructs that are specific to a particular domain, platform, or method. It is not possible to take away any of the constraints that apply to a metamodel, but it is possible to add new constraints that are specific to the profile as given in [30]. Metamodel customizations are defined in a profile, which is then applied to a package. Stereotypes are specific metaclasses, tagged values are standard metaattributes, and profiles are specific kinds of packages.

A. Stereotypes

A UML stereotype is defined in [30] as a new kind of model element defined within the model based on an existing kind of model element. Stereotype is a profile class which defines how an existing metaclass may be extended as part of a profile. It enables the use of a platform or domain specific terminology or notation in place of, or in addition to, the ones used for the extended metaclass as defined in [25]. A stereotype cannot be used by itself, but must always be used with one of the metaclasses it extends. Stereotype cannot be extended by another stereotype. A stereotype uses the same notation as a class, with the keyword « stereotype » shown before or above the name of the stereotype. Stereotype names should not clash with keyword names for the extended model element. Because stereotype is a class, it may have properties. Properties of a stereotype are referred to as tag definitions. When a stereotype is applied to a model element, the values of the properties are referred to as tagged values [13].

The use of this powerful mechanism has both advantages and risks. The advantage is that we can easily create modeling languages for specific application domains with more expressive and precisely defined modeling elements. Examples of domains for which such extensions have already been defined are the real-time [27], the business [28] and the Web domain [29]. The risk is the excessive use of stereotypes that can make a language both difficult to handle and to understand.

Some stereotypes only change the notation of a modeling element; they just serve as a kind of comment. Powerful stereotypes, instead, add or redefine semantic restrictions on the metamodel elements.

B. Tagged Values

Stereotype is applied when it is used on use case diagrams, class diagrams, deployment diagrams, etc. As per the uml diagram of [30] when a stereotype is applied to a model element, the values of its properties may be referred to as tagged values. UML 1.x defined tagged value as one of UML extensibility mechanisms permitting arbitrary information (which could not be expressed by UML) to be attached to models [13]. Tagged value is a keyword-value pair that may be attached to any kind of model element. A UML tagged value is a (tag, value) pair that permits arbitrary information to be attached to any model element. The keyword is called a tag. Each tag represents a particular kind of property applicable to one or many kinds of model elements. Both the tag and the value are usually encoded as strings though UML tool allow to use other data types for values. Tagged value specification in UML 1.x has the form name = value.

Where name is the name of a tag or metamodel attribute and value is an arbitrary string that denotes its value. Tagged value (as well as metamodel attribute) is displayed as a comma delimited sequence of properties inside a pair of curly braces "{" and "}". The information is expressed in text form and is commonly used to store non-functional requirements or project management information. The interpretation of the value is a convention between the modeler and the modelling tool.

C. Constraints

According to [30] UML constraint has been defined as a condition or restriction that allows new semantics to be specified linguistically for a model element. A constraint is a package able element which represents some 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 true or false. Constraint must be satisfied (i.e. evaluated to true) by a correct design of the system. Constraints are commonly used for various elements on class diagrams. For example, operation can have pre-condition and/or a post-condition constraints. Constraint could have an optional name, though usually it is anonymous. A constraint is shown as a text string in curly braces according to the following syntax from [25]:

Constraint: = '{ [ name : ] Boolean-expression }'

UML specification does not restrict languages which could be used to express constraint.
Some examples of constraint languages are: OCL, Java, some machine readable language or a natural language. OCL is a constraint language predefined in UML but if some UML tool is used to draw diagrams, any constraint language supported by that tool could be applied. For an element whose notation is a text string (such as a class attribute), the constraint string may follow the element text string in curly braces.

D. Package
A package in the Unified Modeling Language is defined in [30] and is used "to group elements, and to provide a namespace for the grouped elements". A package may contain other packages, thus providing for a hierarchical organization of packages. Pretty much all UML elements can be grouped into packages [25].

E. Association
Association is a relationship between classifiers which is used to show that instances of classifiers could be either linked to each other or combined logically or physically into some aggregation [30]. UML specification categorizes association as semantic relationship. Some other UML sources also categorize association as a structural relationship or a link [25].

Structural relationship is used for describing links between objects. It may also include labels to indicate number and role of the links. In the example shown below there may be any number of employees (*) each of which has 0 or 1 employer in [25]. Association could be used on different types of UML structure diagrams: Several concepts like Aggregation type, navigability, and end ownership are related to association.

F. Class
The definition of class in [30] says that a class is a classifier which describes a set of objects that share the same features, constraints, or semantics (meaning). A class is shown as a solid-outline rectangle containing the class name, and optionally with compartments separated by horizontal lines containing features or other members of the classifier. Features of a class are attributes and operations. When class is shown with three compartments, the middle compartment holds a list of attributes and the bottom compartment holds a list of operations. Attributes and operations should be left justified in plain face, with the first letter of the names in lower case. Attributes of a class are represented by instances of property that are owned by the class. Some of these attributes may represent the navigable ends of binary associations. Objects of a class must contain values for each attribute that is a member of that class, in accordance with the characteristics of the attribute, for example its type and multiplicity.

Attributes or operations may be grouped by visibility. A visibility keyword or symbol in this case can be given once for multiple features with the same visibility. Additional compartments may be provided to show other details, such as constraints, or just to divide features.

G. Property
Property is a structural feature which could represent an attribute of a classifier, or a member end of association, or a part of a structured classifier of [30]. As a structural feature, property represents some named part of the structure of a classifier[25].

H. Dependencies
Dependency has been described in [30] as a semantic relationship in which a change on one thing (the independent thing) may cause changes in the other thing (the dependent thing). It is a directed relationship which is used to show that some UML element or a set of elements requires, needs or depends on other model elements for specification or implementation. Because of this, dependency is called a supplier-client relationship, where supplier provides something to the client, and thus the client is in some sense incomplete while semantically or structurally dependent on the supplier element(s). Modification of the supplier may impact the client elements. Dependency is a relationship between named elements, which in UML includes a lot of different elements, e.g. classes, interfaces, components, artifacts, packages etc.

Figure 2 UML profile for WebGRL diagrams

VI. THE CONTENT WEBGRL MODEL
The content WebGRL metamodel elements are Actors, Content Goal, Content SoftGoal, task, different types of link, dependency and resources have been defined by us as part of the specific WebGRL model explained in our paper [9] on specific WebGRL.
The Content WebGRL diagram represents the content requirements of the web application. We further transform this using the Enhanced A-OOH design models as intermediary to map the WebGRL content diagrams into a UML Profile as explained below. The EA-OOH approach used by us is explained in [22]

A. The Method:

All content goals in the WebGRL diagrams are represented as Content Nodes representing Domain Classes for the resource used by them to satisfy that goal. The goal always states a function which is to be carried out. That function may be a service task or a navigation task. If it leads to navigation to another content class then it is a navigation task. However, if it requires performing an operation on the attributes within the domain class defined for that content goal then it is a service task. The operation within the domain class would be performed on some content variables or patterns within the domain class itself. These variables or content pattern are to be represented as the attributes of the domain class with operations in the domain class defined on them. If the operation to satisfy this goal needs a navigation to obtain information from another class then a relationship is defined between the two domain classes. Softgoals are represented as conditions of that domain class defined as member conditions on the model elements, i.e., domain class, association or conditions on the attribute of the domain class with satisfaction levels. All these elements are represented in the UML Profile and UML Metamodel for the Content webGRL in the figure 3 below.

TABLE I
UML METACLASS FOR WEBGRL CONTENT

<table>
<thead>
<tr>
<th>WebGRL Content</th>
<th>Domain</th>
<th>Stereotyped UML Metaclass</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model elements</td>
<td>Model</td>
<td>Stereotypes</td>
</tr>
<tr>
<td>GRL spec</td>
<td>Content Model</td>
<td>Model</td>
</tr>
<tr>
<td>GRL model</td>
<td>Content model</td>
<td>Named Element</td>
</tr>
<tr>
<td>Element</td>
<td>Domain model</td>
<td>Class</td>
</tr>
<tr>
<td>Actor</td>
<td>Domain Class</td>
<td>Class</td>
</tr>
<tr>
<td>Intentional Element Goal &amp; Resource</td>
<td>Condition</td>
<td>Constraint</td>
</tr>
<tr>
<td>Intentional Element Softgoal</td>
<td>Domain Class</td>
<td>Operation</td>
</tr>
<tr>
<td>Element Task</td>
<td>Operation</td>
<td></td>
</tr>
<tr>
<td>Element Link</td>
<td>Link</td>
<td>Relationship</td>
</tr>
<tr>
<td>Contribution</td>
<td>Relationship</td>
<td>Directed Relationship</td>
</tr>
<tr>
<td>Type Contribution</td>
<td>Enumeration</td>
<td>Type attribute</td>
</tr>
<tr>
<td>Decomposition Type</td>
<td>Association</td>
<td>Association</td>
</tr>
<tr>
<td>Dependency</td>
<td>Association</td>
<td></td>
</tr>
</tbody>
</table>

VII. DERIVING THE NAVIGATIONAL MODEL

The Navigation WebGRL Diagram has elements like navigation goal, navigation tasks, softgoals, navigation links, contribution links, decomposition links and dependency. We transform these elements using the Navigation WebGRL Diagram as input to generate the Navigation model and the NAD of EA-OOH as intermediary design models which are supported by a UML Profile.
The Navigation Model of EA-OOH describes a navigation view on data specified by the Domain Model in OO-H the NM is captured by one or more Navigation Access Diagrams (i.e. NADs). The A-OOH Navigational Model (NM) has been enhanced to transform the WebGRL navigation diagram. The 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, tours and Guided Tours or queries which collaborate in the fulfilment 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 enhance the navigation links to represent the contribution link and decomposition links which will be represented as associations in the navigation model.

A. The Method:

To derive the NM we take into account the navigation goals. All navigation goals in the WebGRL diagrams are represented as Navigation Classes for the resource used by them to satisfy that goal. They are derived from the domain classes of the Domain model. The navigation goal always states a navigation task that leads to navigation to another content class then a traversal link is added between the two navigation classes that is supported by the access primitives depending on the requirement stated in the goal.

Softgoals are represented as conditions of that navigation class defined as member conditions on the model elements, i.e. navigation class, association or conditions on the attribute of the navigation class with satisfaction levels. The UML Profile for the navigation WebGRL is shown in figure 4 below.

The NAD Navigational Node is defined as an extension of the UML class concept which has attributes and operations (also extensions of the UML concepts). There have been defined different stereotypes for representing the different types of Navigational Nodes (i.e. <<NavigationalClass>>, <<Menu>>, <<Index>>, <<Guided Tour>>, <<Query>> and <<Showall>>).

The Navigational Class concept has information about the name of the root concept in the Domain model (stored in the tagged value rootConcept).

![Figure 4 UML profile for navigation WebGRL](image-url)

As aforementioned a profile for the NAD has been defined. Extending the UML concepts of class, association, constraint and tagged values the NAD concepts are specified. The purpose of defining an UML profile is to provide an easy mechanism of adaptation to the UML metamodel to elements that are specific of a particular domain, platform or method. In this sense, the particular profile for the NAD consists in adapting the elements defined in the NAD to the UML metamodel.
### Table II

<table>
<thead>
<tr>
<th>WebGRL Model Element</th>
<th>Navigation Model Stereotype</th>
<th>Stereotyped UML Metaclass</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebGRL spec</td>
<td>Model</td>
<td>Model</td>
</tr>
<tr>
<td>Actor</td>
<td>Navigation Class</td>
<td>Class</td>
</tr>
<tr>
<td>Intentional Element Goal &amp; Resource</td>
<td>Navigation Class</td>
<td>Constraint</td>
</tr>
<tr>
<td>Intentional Element Softgoal</td>
<td>Navigational Condition</td>
<td>Class</td>
</tr>
<tr>
<td>Intentional Element Task</td>
<td>Navigational Class</td>
<td>Class</td>
</tr>
<tr>
<td>Importance Type</td>
<td>Navigational Importance</td>
<td>Enumeration</td>
</tr>
<tr>
<td>Element Link</td>
<td>Link</td>
<td>Association</td>
</tr>
<tr>
<td>Contribution Type</td>
<td>Contribution type</td>
<td>Enumeration</td>
</tr>
<tr>
<td>Decomposition Type</td>
<td>Decomposition type</td>
<td>Enumeration</td>
</tr>
<tr>
<td>Dependency</td>
<td>Primitive type</td>
<td>Dependency</td>
</tr>
</tbody>
</table>

#### VIII. DERIVING THE PRESENTATION MODEL

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 A-OOH based models we have enhanced the existing A-OOH DPD to the Enhanced Design Presentation diagram (i.e. EDPD). The enhanced version EDPD and its modelling elements are described in [23].

Using the Presentation WebGRL Diagram as input having elements like presentation goal, navigation tasks, softgoals, navigation links, contribution links, decomposition links, dependency. We transform these elements using the Enhanced DPD as intermediary into a UML Profile.

**A. The Method:**

1. To derive the EDPD we take into account the main presentational goal first. This is represented by the Main or Home page of the website with Frames, Frameset and Presentational Class with Interface Components to represent the content of the Home Page and have navigation links represented by Interface Component of anchored collection to other Presentation Goals represented by Presentation Pages. All Presentation Goals in the Presentation webGRL diagrams are represented as Presentation pages with Presentation Classes. There is one Presentation Classes for each Navigation Class in The NAD and Index.

2. We travel from the main presentational goal to other presentational goal by adding an Interface Component like anchor or menu item to navigate between the presentation goals now represented by Presentation Pages. The content for each Presentation Goal is displayed by Presentation Class and its Interface Components on that Presentation Page.

3. We move from top to bottom by creating a Presentational Page for each Presentation Goal with a navigation link to contain or navigate to other pages.

4. In case of a presentation Goal with requirements for form the Presentation Page must have a display link to the Interface Component Form with submit link. A list or an index is an anchored collection navigating to other Presentational pages.

5. Presentation Goal with Search use build to generate the new Presentation Page each time a search is performed with the help of server pages.

6. Soft goals are represented as conditions of that presentation class of the presentation page to which they are attached. They are defined as member conditions on the model elements, presentation class, association or conditions on the attribute of the presentation class with satisfaction levels. The zero level of the EDPD shows how the navigation nodes are grouped into abstract pages (and page chunks in the case) in the Website.
In this case, each of the Navigational Nodes correspond with one Presentation Page, with the presentation class corresponding to its resource or the content class and the information on the page is represented with the help of Interface Components.

Once the DPD MOF metamodel has been defined, a UML profile is presented from the DPD model for expressing it in UML 2.0. We present the UML Profile for the EDPD below with their UML extensions in figure 5. The Structure Nodes defined as an extension of the UML class concept are Window, Frame and Frameset, Layout, Interface Component, Form, Presentation Class and cell. The other two Structure Nodes Presentation Page and Page chunk are defined as an extension of the UML Package concept. Each of these concepts is represented by means of a stereotyped class.

The Interface Components Anchor, Image, text, anchored Collection and Collection also extends from the UML class concept. Analogous to the Interface Components, Forms also extend from the UML class concept and its elements extend from the UML attributes concept. The DPD Presentation Links IncludeFrame and IncludeCell are defined as an extension of the UML Association metaclass. The IncludeFrame link is represented with the stereotype <<IncludeFrame>> and the IncludeCell link with the stereotype <<IncludeCell>>.

The DPD Presentation Links Navigates and Displays are defined as an extension of the UML Association metaclass. The Navigate link is represented with the stereotype <<Navigate>> and the Display link with the stereotype <<Display>>. The rest of the Presentation links Contains, Builds and Submit are defined as an extension of the UML dependency metaclass and represented with the stereotype <<Contains>>, <<Builds>> and <<Submit>> respectively.

Table III

<table>
<thead>
<tr>
<th>WebGRL Model Elements</th>
<th>Presentation Model Stereotype</th>
<th>Stereotyped UML Metaclass</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) WebGRL spec</td>
<td>1) Model</td>
<td>1) Model</td>
</tr>
<tr>
<td>2) Actor</td>
<td>2) Presentation Page</td>
<td>2) Package</td>
</tr>
<tr>
<td>3) Intentional Element Goal &amp; Resource</td>
<td>3) Presentation Class</td>
<td>3) Class</td>
</tr>
<tr>
<td>4) Intentional Element Softgoal</td>
<td>4) Presentation Conditional Condition</td>
<td>4) Constraint</td>
</tr>
<tr>
<td>5) Intentional Element Task</td>
<td>5) Presentation Class-Interface Components</td>
<td>5) Class</td>
</tr>
<tr>
<td>6) Importance Type</td>
<td>6) Presentation Importance</td>
<td>6) Enumeration</td>
</tr>
<tr>
<td>7) Element Link</td>
<td>7) Link</td>
<td>7) Association</td>
</tr>
<tr>
<td>8) Contribution</td>
<td>8) Navigational Link</td>
<td>8) Association</td>
</tr>
<tr>
<td>9) Contribution Type</td>
<td>9) Contribution type</td>
<td>9) Association</td>
</tr>
<tr>
<td>10) Decomposition</td>
<td>10) Navigational Link</td>
<td>10) Association</td>
</tr>
<tr>
<td>11) Decomposition Type</td>
<td>11) Decomposition type</td>
<td>11) Enumeration</td>
</tr>
<tr>
<td>12) Dependency</td>
<td>12) Builds</td>
<td>12) Dependency</td>
</tr>
</tbody>
</table>
IX. CONCLUSION

A UML Profile for the WebGRL specific model transformation approach extending the GOREWEB framework from the requirements phase to the design phase has been presented in this paper. The user goals can be both hard goals and soft goals; hence we need an approach that models the softgoals as well as web specific goals in the design phase. By applying the model transformation approach stated above we capture the goals as well as softgoals in the requirements phase and seamlessly transfer them to the design models suited for web applications along with a UML compliant UML profile to support them. The UML Profiles for presented in this paper are namely the Domain Model, Navigation Model and the Presentation Model. In future, we propose to enhance the A-OOH design model to incorporate the presentation, adaptation and business process models with UML profile to support them. This would reduce the probability of risks and improve the quality of the product while keeping the stakeholders’ goals in mind.

REFERENCES


