AN AUTOMATED TRANSFORMATION FROM REQUIREMENT TO DESIGN MODEL FOR WEB APPLICATIONS

Sangeeta Srivastava  
Department of Computer Science, Bhaskaracharya College of Applied Sciences, University of Delhi, Delhi, India.  
E-mail: sangeeta.srivastava@bcas.du.ac.in

Anuja Soni  
Department of Computer Science, Deen Dayal Upadhyaya College, University of Delhi, Delhi, India.  
E-mail: anujasoni@ddu.du.ac.in

Vibha Gaur *  
*Corresponding Author  
Department of Computer Science, Acharya Narendra Dev College, University of Delhi, Delhi, India.  
E-mail: vibhagaur@andc.du.ac.in

ABSTRACT
Web applications dominate all aspects of our lives and hence their systematic development has become an absolute necessity. Though a multitude of web engineering approaches is available, the focus is mainly on their presentation and implementation. The requirements and design phases for web application development are often overlooked. This paper presents an automated Web-Application Development (WAD) tool that captures the users' requirements with the Web-specific Goal-Oriented Requirements Language (WebGRL) diagrams and produces an Enhanced Adaptive Object-Oriented Hypermedia (EA-OOH) design model. The transformation tool from WebGRL to EA-OOH design model is exclusive to web applications and provides a Unified Modelling Language (UML) Compliant UML Profile, for an easy adaption from a platform-independent model to a platform-specific model that facilitates the implementation. The proposed WAD tool eases the job of the web-engineers to a very large extent and also reduces the development time.

KEYWORDS
Content Model, Presentation Model, Navigation Model, Web Engineering, UML, EA-OOH Design Model.

1. INTRODUCTION
Web Applications and services have embraced almost all spheres of our lives and have become crucial (Ceri et al., 2007) and (Cachero & Gomez, 2002) for existence. However, the development of web applications is not usually done systematically. Many times, even the methodical approach of Software Development Life Cycle (SDLC) is not followed completely (Schwabe and Rossi, 1998), for example, the requirements and design engineering steps are overlooked or rushed through, resulting in an unstructured application that may fail to achieve its primary goal.  
Software development is viewed as a transformation function that converts requirements into the desired solution, with the development team focusing primarily on proper planning and developing the transformation function. The function-oriented structured design process is heavily reliant on functional abstraction and decomposition. Web applications, on the other hand, require the stylish, useful, financial, and socio-political dimensions of the design process. It might include extensive modeling and intuitive design solutions while the conventional software development of information
systems using databases, does not help in facilitating the hypertext metaphor. Hypermedia application development requires web-specific processes viz. the Conceptual Design, the Navigation Design, the Presentation Design, and Implementation which are performed as a combination of different styles of development. At every step, a design model is constructed or enhanced. Therefore, Web Engineering has evolved to address the specific needs of web applications. However, there have been a few recent attempts to encourage web application development using a model-driven methodology to allow software developers to concentrate on problem domain-specific modeling and analysis, as well as structured software recycling. Therefore, a disciplined and systematic approach for the development of web applications is needed (Schwabe et al., 2004), and (Kraus, 2007). As a wide variety of stakeholders and platforms are involved in web applications, the requirement and design phases for these applications are different from traditional software development. Navigational and Presentation issues are often not emphasized in conventional software development, while these are very crucial aspects for a web application. A web-specific methodical development process exclusively for web applications is envisaged in this work that automates the transformation of the requirement model into a design model.

To build a high-quality and consistent design model using a methodical and well-planned web engineering methodology, this work utilizes the Web-specific Goal-Oriented Requirements Language (WebGRL) diagrams to capture the requirements for a web application comprehensively and a novel improvised Enhanced Adaptive Object-Oriented Hypermedia (EA-OOH) design model introduced in our previous work (Srivastava, 2015) is adopted, which has the potential to display the various dimensions of a web application in an all-encompassing manner. Three web-specific design models viz. Content, Navigation, and Presentation are developed in the EA-OOH design model. Formal modeling of web applications is based on these web-specific models. Therefore, these web-specific design models are utilized to assist in the transition from the requirements phase to the design phase and the implementation phase. The EA-OOH model's output is UML compliant and can be coded in an Object-Oriented Language.

This paper presents a structured Requirements to Design Transformation using a Web Application Development (WAD) tool that reduces the work of the design engineer to a large extent and is unique to web applications. WebGRL diagrams and associated modeling elements are used as input and are turned into EA-OOH Design Models by employing transformation algorithms for the automated development of EA-OOH design (Srivastava, 2014b). Different WebGRL diagrams can be used as input and evaluated for the choice of alternative solutions, by using the automated transformation tool to generate alternate designs so that appropriate design rationale is there and the satisfaction of the stakeholders from the anticipated solution can be estimated. The salient features of the proposed WAD tool are listed below:

1. The transformation tool is comprehensive as it comprises both explicit and implicit requirements as input, which is typically overlooked by other web application development design models.
2. This tool is unique to web application development, as it employs web-specific requirements and design features that are distinct from those used in traditional software development.
3. Even though the WAD tool caters to web application development, it adheres to time-tested software engineering principles for the systematic development of a web application, which is mostly done in an ad-hoc manner due to the paucity of time (Chawla et al., 2011).
4. The proposed automation tool is also user-friendly and supportive for the comprehensive transformation process from the WebGRL requirements model to the EA-OOH design model that may be utilized by the web application development industry to achieve structured applications and manage them efficiently. It also helps design engineers in evaluating the different design alternatives in a very short period.

The organization of the paper is as follows: Related Work in the area of web application development is presented in section 2 while Section 3 describes the development cycle of the proposed WAD tool. Section 4 illustrates an application of the WAD tool to transform the Requirements model to the EA-OOH Design model with the help of a common example. Section 5 concludes the paper.
2. RELATED WORK

Web Engineering is a process of developing Web applications (Koch & Wirsing, 2001), (Koch et al., 2008), (Koch et al., 2012), (Koch & Kozuruba, 2012) that emphasize a methodical, efficient as well as personalization, and semi-automatic generation founded on UML and the mechanism for UML extension. Modern web applications can be broadly classified into four categories based on their features viz. Semantic Web Applications (SWA); Intelligent Web Application (IWA); Ubiquitous Web Applications (UWA); and Rich Internet Applications (RIA) (Wakil & Jawawi, 2017) Interaction Flow Modelling Language Meta-model is a defined standard for web engineering for ubiquitous Rich Internet Applications (Wakil & Jawawi, 2018)

A wide variety of approaches for web application development have been proposed. One such popular approach presented by Schwabe et al. is the Object-Oriented Hypermedia Design Model (OOHDM) (Schwabe & Rossi, 1995), (Schwabe & Rossi, 1998), and (Schwabe et al., 2004) and is based on the Object-Oriented paradigm and the UML-based Web Engineering (UWE). Most of these methodologies emphasize the structured development of web applications. The analysis of existing methodologies reveals that the researchers agree on the meaning of a "structure model" as a "navigation model", and a "presentation model".

Unified Modeling Language (UML) is a widely accepted standard for modeling software and is used by most approaches for web application development as it is easier to learn (UML, 2001). The Object-Oriented Hypermedia Design Method (OOHDM) allows the succinct specification and implementation of hypermedia applications. The objects, their naming, and the interpretation are no longer tied to semantics. This is achieved by applying different models representing the application's knowledge containing conceptual, navigation, and configuration aspects.

Though the web engineering approaches follow a structured methodology, however, they tend to focus more on design and implementation. To understand and improve the web development process it is imperative to take up the requirement analysis phase thoroughly. Also, developers do not cover the implicit requirements of many stakeholders involved while designing the web application. It becomes necessary to understand requirements from all sources exhaustively before starting the designing and implementation of the web application for a good quality product. An automated approach from the requirement to design phase using a WAD tool is envisioned in this work.

The proposed WAD tool is exhaustive and follows a more holistic approach. It focuses deeply both on requirement elicitation and analysis, and transformation to the design model. Automation of the entire process from the requirement to the design model makes it cost-effective as the development time of the application reduces considerably. The following section describes the proposed tool for web applications.

3. DEVELOPMENT LIFE CYCLE FOR WEB APPLICATION DEVELOPMENT (WAD) TOOL

The proposed Requirements to Design Transformation tool utilizes the Enhanced Adaptive Object-Oriented Hypermedia (EA-OOH) model (Schwabe & Rossi, 1998) that is based on the Adaptive Object-Oriented Hypermedia (A-OOH) model (Koch and Wirsing, 2001). The EA-OOH model takes the following workflows into account:

a) Requirement Analysis and Modeling: The requirement modeling phase involves capturing users’ functional as well as non-functional requirements for a web application using the Goal-Oriented Requirement Engineering for Web Applications (GOREWEB) framework described in our previous work (Chawla et al., 2014, 2016). In phase one, the "Base WebGRL diagram" is produced of the web application under development. In the second phase, the "Base WebGRL diagram" is extended into the "Specific WebGRL diagrams" that comprise the Content, Navigation, and Presentation WebGRL diagrams. These three specific WebGRL diagrams are used as the input to the design phase. The WebGRL diagrams capture not just functional
requirements but non-functional requirements as well. Also, these WebGRL diagrams are tuned specifically for web application development.

b) Software Design: The activities of the design phase for the WAD tool are briefly explained below:

- Domain Design Model: The “Content WebGRL diagram” from domain analysis is refined with additional goals and soft goals and the final refined WebGRL diagram is converted to the EA-OOH Domain Design model. This improved Content Webgrl diagram has been used in the model transformation strategy described in our earlier work (Srivastava, 2014b) for a seamless transition into the EAOO-H domain model of the content Webgrl diagram.

- Navigation Design Model: The “Navigation WebGRL diagram” obtained from the requirements phase is used to obtain EA-OOH Navigation Design Model. It comprises various navigational views and paths that are captured using Navigation Access Diagrams explained in our previous work (Srivastava, 2014b).

- Presentation Design Model: The “Presentation WebGRL diagram” involving the location, appearance, and supplementary graphical components of the interfaces is used as input to obtain the EA-OOH Presentation Design model. It shows the information and navigation of each of the abstract interfaces (Srivastava, 2014a).

The “EA-OOH Design Model” thus obtained is represented as a UML compliant class diagram which is the enhancement and development of the “UML profile” (Srivastava, 2015).

c) Implementation: The final workflow considered in the EA-OOH design model is the implementation, where the final WAD tool is developed. The automation of transformation from WebGRL Requirements diagram to EA-OOH Web-specific Design Model as output is implemented as a software framework using Visual Basic (VB) which is an event-based programming language. VB helps to create the EA-OOH design modeling elements and facilitates the transfer of information between the specific WebGRL diagrams to the EA-OOH Design models.

d) Testing: The testing of the Web Application Design Output is performed to ascertain that the final EA-OOH design model includes all the requirements of the web application as intended.

The primary objective of this workflow is to make sure that the implementation is done correctly. In the first two steps, WebGRL diagrams and the domain analysis products viz. Web Specific GRL diagrams are used. In the next step, the output design model is produced by translating these specialized requirements analysis models to their respective design models using the EAOO-H design model. The three specific EA-OOH Design Models are presented briefly in the following section.

4. AN AUTOMATED TOOL FOR REQUIREMENTS TO DESIGN TRANSFORMATION

There is an imminent and compulsive need for first capturing the requirements comprehensively and further supporting the transformation of these requirements to the design phase for web applications seamlessly without losing any of any data presented in the requirements engineering phase. This work proposes a Web Application Development (WAD) tool to automate the transformation of Web Requirements to Design Model.

Figure 1. Schematic Diagram of WAD Tool
The schematic diagram of the WAD tool is presented in Figure 1. The tool takes WebGRL diagrams obtained from the Requirement Analysis phase as an input to produce EA-OOH Design Model as an output. The proposed WAD tool for transforming the WebGRL diagrams into the EA-OOH design model elements is implemented using the Visual Basic language. The screenshot of the GUI of the WAD tool is shown in Figure 2.

Figure 2: Screenshot of GUI of the WAD Tool

A WebURN (User Requirements' Notation for Web) tool (Chawla et al., 2014) is employed for capturing the requirements for the WebGRL framework which guides and produces an output consisting of alternative web-specific WebGRL diagrams (Chawla et al., 2011). These diagrams provide input to the proposed WAD tool and enable the transformation of WebGRL modeling elements into the enhanced notations for Web-specific EA-OOH design models. The WAD tool helps in automating the entire transformation process of identifying the modeling elements of the EA-OOH design model and also helps in the creation of the transformed design model based on the specifications captured at the requirements stage using all the three-web specific WebGRL models.

The proposed WAD tool comprises three modules: The Domain, Navigation, and Presentation transformation modules. The working of all three modules is explained in subsequent sub-sections using an everyday example of the Online Bookstore System. The primary requirements of the Bookstore System as elicited from the stakeholders are listed below:

- Sale of Books (Parent goal)
- Increase profit
- Increase Reachability
- The refined Sub goals for a parent goal: ‘Sale of books’ are listed below:
  - Providing detailed book information (Content goal)
  - Facilitating payments (Navigation goal)
  - Maintaining client/customer details (Content goal)

The Soft Goals gathered from the stakeholders are listed below:

- Provide Secure Transactions
- Improve browsing experience
- Easy to maintain the book store
- Retain old clients/customers
- Attract new clients/customers

The Primary Goals of the Online Bookstore are shown in the screenshot of the Base WebGRL in Figure 3. If we look at the Base WebGRL diagram closely we can see that the goal of “providing detailed book information” is satisfied if its sub-goals viz. "provide a smooth search experience"
(navigation goal), "provide a subject-wise list" (navigation goal), and "present information properly" (presentation goal) is maintained. The navigation goal to "provide a smooth search experience" is split further into sub-goals to “allow book browsing”, “query to search based on the title” and “query to search based on the author”, for the resource “book”. Likewise, the goal “Add to cart” is decomposed into “add a book to cart” and “remove a book from cart” etc. which is related to the resource “cart”. Similarly, other main goals are also divided into detailed sub-goals.

These goals are represented in the base WebGRL diagram as shown in Figure 3, which is refined to give the web-specific WebGRL diagram. The Base WebGRL is prepared by the requirements engineer using the WebURN tool and the goals, sub-goals, and the inter-relationship between different goals is presented in the base WebGRL diagram for the Online Bookstore as shown in Figure 3 that is taken as the Input of Main Menu of the WAD tool.

![Figure 3. The Main Menu with Input of Base Web GRL Diagram for Online Bookstore](image)

The different types of goals for Content, Navigation, and Presentation are denoted by ovals with a notation ‘C’ for Content and ‘P’ for Presentation, and ‘N’ for Navigation Goals (Chawla et al., 2011) in the base WebGRL diagram. The base WebGRL diagram is refined further to get the web-specific WebGRL diagrams namely the content, navigation, and presentation WebGRL diagrams. The EA-OOH design model consists of Domain, Presentation, and Navigation design models. The activities for the development of these web-specific design models are briefly explained in the following subsections. The three transformation modules take the WebGRL diagrams as an input to transform and generate the respective design models. The transformation modules with input, the algorithm, and the output are discussed as follows:

### 4.1 Domain Transformation Module and the EA-OOH “Domain Model (DM)”

The Content WebGRL diagram is made up of content goals, soft-goals, relationships, and resources associated with these content goals (Chawla et al., 2014, 2016). These goals are used to store the information of the web application for transformation into the domain classes, relationships, cardinalities using the following transformation process. The snapshot of the Content WebGRL submenu is presented in Figure 4 that is used as an input for the Domain Transformation Module. The actors identified in the requirements workflow of the WebGRL content model are transformed into EA-OOH domain classes to store information of the web application under development.

The EAOO-H Domain Model is a class diagram that complies with UML. It represents the framework's static component and provides the key points of the structure and functionality expected of the application's related concepts. A class diagram's key model elements are the classes, attributes, conditions, and procedures, as well as their relationships. Though the key concepts are the same as in the A-OOH domain model, the interpretation and semantics of the domain model concepts vary significantly. The Domain module of the proposed tool takes the first step towards the transformation from the requirements to the design model i.e., the transformation of the Content WebGRL diagram...
into the EA-OOH Domain design model.

Figure 4. The screenshot of the input Content WebGRL Diagram for Online Bookstore

The Domain Transformation Module utilizes the given input, determines the association and hierarchical relationships between the domain classes, and the most relevant attributes and operations of a given domain class. To ensure that this domain model is a complete representation of the content WebGRL diagram after transformation, a set of transformation rules are defined (Srivastava, 2014a) and (Srivastava, 2014b) along with a Transformation Algorithm 1 presented in Figure 5. The algorithm used for transformation for Content WebGRL Diagrams to EA-OOH Domain Design model uses the following assumptions:

- A is a parent/primary goal with the collection of links to sub-goals starting from level 1
- Xi are sub-goals of A for i=1 to n
- Si are soft goals for i=1 to m
- Ri are resources for i=1 to p
- Li are the navigation links attached from A to X for i=1 to q
- DCi (Ri) are the domain classes for goals attached to a resource R for i=1 to i
- Mi are means-end tasks attached to a goal which will be presented as an operation for the domain class DCi

| Algorithm 1: Algorithm for Transformation of Content WebGRL to EA-OOH Domain Model |
|---------------------------------|----------------------------------|
| **Input**: Content WebGRL Diagram | **Output**: EA-OOH Domain Design Model |
| **Do for each level i=1 to n //Xi is the primary parent goal** |
| **Case 1**: for each Xi linked directly to a Ri // Xi+1 is the sub-goal |
| create DCi = " Ri " |
| // all resources connected directly to goals are transformed to respective domain classes// |
| **Case 2**: for each (Xi that is not directly linked to a Resource but subdivides into (Xi+1) subgoals which are linked to the Resource) |
| Check the links between the parent goals and subgoals |
| { |
| **If** (the link is of type generalized) **Then** |
| create Xi =Generalised DCi and Xi= DCi (Ri) with Li as the association |
between Xi and X;
   // Xi is a generalized domain class with specialized domain sub-classes
   
   **If** (the link is of type aggregate) **Then**
   create Xi =Aggregate DCi and Xi= DCi (Ri) with Li as the association between A and X;
   // Xi is the aggregate domain class composed of domain classes linked to resources
   
   **If** (the link is of type simple decomposition) **Then**
   create DCi=" Ri " and add association links between parent DCi and decomposed DCi;
   // There is a simple relationship between parent and decomposed domain classes
   }

   **Case 3**: Xi goals that terminate into means-end task Mi
   Xi=DCi(Ri).operation(Mi))
   // If a goal is satisfied by a function, then it’s embedded as a method of that domain class

   **Case 4**: (Xi is a navigation goal which is a navigation task between A and Xi) then
   Li is represented as an association between A and Xi
   // Each navigation goal/link is transformed into an association between the respective domain classes

   **Endcase**

   **For** each Si attached to a goal A or Xi =Constraint on the DC(Ri) or its members **Endfor**
   // Each soft goal on a parent or sub-goal is transformed into a constraint on the transformed domain class or its members

**Enddo**

Figure 5. The Content WebGRL to EA-OOH Domain Model Transformation Algorithm

![Figure 5](image-url)

Figure 6. The Screenshot of the Output EA-OOH Domain Design Model for the Online Bookstore

The screenshot of the output EA-OOH domain design model for the Online Bookstore representing the various domain Classes such as the Book, Author, Cart, Customer, Order, Category, and Transaction is presented in Figure 6. The EAOO-H Domain Model obtained after the transformation from Content WebGRL specifies the structure of domain data that includes a class diagram with classes, their
attributes, conditions, and procedures, as well as their relationships. The model is UML compliant and summarizes the layout and functionality of the web application’s related concepts.

4.2 Navigation Transformation Module and the EA-GOH Navigation Model (NM)

The Navigation module is the second step towards the transformation from the requirements to the design model i.e., the transformation of the Navigation WebGRL diagrams into the EA-GOH Navigation Design model. The input consisting of the Navigation WebGRL diagram and its modeling elements are entered using the Navigation WebGRL SubMenu.

The Navigation WebGRL diagram consists of navigation goals, navigation links, and associations between these goals. These navigation goals are used to represent the navigational information comprising of navigation routes for traversing the web application. The information comprises various navigational views and paths that are captured using Navigation Access Diagrams (Srivastava, 2014b). To ensure that this navigation model is a complete representation of the Navigation WebGRL diagram after transformation, a set of transformation rules presented in our earlier work are utilized (Srivastava, 2014a) and (Srivastava, 2014b).

The Navigation model transformation algorithm is used for transforming the Navigation WebGRL diagram to the EA-GOH Navigation Design Model. The primary input for the design navigation activity is the establishment of the different navigational paths, which should be specified for the varying functional requirements. These navigational paths lead to the organization of the information into separate abstract pages. Finally, the EA-GOH Navigation design model specifies the navigation classes, their association, and the corresponding navigation links.

![Figure 7. The Screenshot of the Input Navigation WebGRL Diagram for the Online Bookstore](image)

The WAD Navigation module automates the navigation design model transformation process. To transform the WebGRL navigation model, the navigation classes are identified. The associations between these navigation classes and the links are determined to identify the path to be followed by the user based on the conditions associated with these navigation classes and links. There are various types of navigation links that need to be modeled differently. The most relevant attributes and operations for a given navigation class also need to be identified.

A Navigation Transformation Algorithm 2 is used for the conversion of the WebGRL Navigation model to the EA-GOH Navigation design model. The input consisting of the Navigation WebGRL diagram and its modeling elements are entered using the Navigation WebGRL SubMenu. The snapshot of the same for Online Bookstore is shown in Figure 7. In the Navigation WebGRL diagram,
the main navigation goal “allow smooth searching options” is split into sub-goals: “query to search the book using the title” and “query to search the book using author”. Likewise, the navigation goal “provide cart” is split into “add a book to cart”, “continue browsing”, “checkout”, and “remove a book from cart”. Similarly, the goal “proceed for payment” is decomposed into “place an order”, and “pay now”. The analysis of the Navigation WebGRL Diagram is used for the transformation of these navigation goals and links into the EA-OOH Navigation Design Model using the Transformation Algorithm 2 and the output is generated automatically by the WAD tool.

The Navigation Design Model Transformation Algorithm 2 is given below in Figure 8. The pseudo-code and the terminology used for defining the navigation WebGRL diagrams is as follows:

- A is a parent goal
- Xi are navigational sub-goals i=1 to n attached to a Content goal/Business goal/Presentation goal
- Si are soft goals i=1 to m
- Ri are resource i=1 to r with resource name R
- AX(Li) are the navigation links attached from A to X for i=1 to n;
- AX(Ti) are the traversal links attached from A to X for i=1 to m;
- AX(Si) are the service links attached from A to X for i=1 to p;
- Mni are the menus for navigation goals with a collection of links for i=1 to q;
- APi are the Access Primitives for navigation goals with some operations referring to an existing NCi
- Li are the Index access primitives for i=1 to l
- SAi are the Showall access primitives for i=1 to m
- GTi are the guided tour access primitives for i=1 to p
- Qi are the Query access primitives for i=1 to q
- NCi(Ai) are the attributes for the Navigation Class for i=1 to l
- NCi(Oi) are the operations for the Navigation Class for i=1 to m
- Mi are means-end tasks attached to a sub-goal or a parent goal which will be stored as the operation Oi for i=1 to m for the parent or sub-goal transformed into a navigation class.
- NCi are the navigation classes for each domain class for i=1 to m.
- DCi (Ri) are the domain classes for goals attached to a resource R for i=1 to p.

### Algorithm 2 Algorithm for Transformation to EA-OOH Navigation Model

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>For the main parent goal A, create a homepage with Main Menu M1 with a collection of navigation links AX(Li) to the subgoals Xi connected to it at one level below; // Create the Main Menu on the Homepage// Endfor</td>
<td></td>
</tr>
</tbody>
</table>
| For each DCi in the Domain Model create a corresponding NCi with a navigation link from the Main Menu to display the Content on a Navigated web page; Endfor // Create Navigation Classes corresponding to each Domain Class for content presentation soft goal// Do for each level i=1 to n // First Check the type of links if they are simple traversal, decomposition into sub-menu or means-end task service link. This step checks whether there are submenus or simple links from the Main menu or links to operations embedded within the Homepage or the Parent webpage// Start at level 1 and for each Xi, sub-goal linked to the parent goal A check the link connecting A to Xi and Xi to Xi+1 For each navigation link from A to its subgoal Xi Case 1: (AXi is a decomposition link) then check the number of links and for each link
repeat step 4 case 2 or case 3 to determine the type of link and decrement the number of links till zero.

// Repeat step 6 to check the type of each of the decomposed links if they are simple traversal, decomposition into sub-menu or means-end task service link and go the next level. //

Case 2: (AX(Li) is a simple navigation link)
then {Ax(Li)= AX(Ti) between Ai and Xi (where Xi already exists as some NCi corresponding to a DCi else Create an NCi corresponding to Xi with an Access Primitive APIi)} then go to Step 5.

// Each Navigational Class NCi existing or created has an Access Primitives associated with it for satisfying the navigation goals operation//

Case 3: (AX(Li) is a Means-End Task link between Ai and X) where Ai maybe some NCi with a Means-end task link to Xi) then {AX(Li)=AX(Si) and Mi=Ai (Oi)}
// Each Navigational Means-End Link provides a service by associating the operation within the Means-end task with the existing Class NCi corresponding to the parent goal //

Endcase
Endfor
For each Ai = API check each Access Primitive type

Case 1: (AX(Li) between API and NCi is a simple traversal link with NCi=list of Ai Then
{API=Ii}
// Ii are the Index access primitives for i=1 to l//

Case 2: (AX(Li) between API and NCi is a simple traversal link with NCi(Ai)=ordered list of Ai Then (API=GTi)
// GTi are the guided tour access primitives for i=1 to p//

Case 3: (AX(Li) between API and NCi is a simple traversal link) Then (API=Sai)
// SAI are the Showall access primitives for i=1 to m//

Case 4: (AX(Li) between API and NCi is a simple traversal link with NCi(Ai)=query string and NCi(Ai)=ordered list) Then (API=Qi and GTi with traversal link between Qi and GTi)
// Qi are the Query string access primitives with Guided tour for i=1 to q//

Case 5: (AX(Li) between API and NCi is a simple traversal link with NCi(Ai)=query string) Then {API=Qi and SAI with traversal link between Qi and SAI}
// Qi are the Query string access primitives with the Showall access primitives for i=1 to m //

Endcase
Endfor
Enddo

Figure 8. The Navigation WebGRL to EA-OOH Navigation Model Transformation Algorithm

The automatic transformation is very helpful for the design engineer for choosing between the various alternatives as it saves both time and cost. The EA-OOH Navigation design model for the Online Book Store consists of Navigation classes and their association as shown in Figure 9. The Main Menu navigation class of the Online Bookstore and the navigation links from this Main Menu navigation class to the other navigation classes like “query to Author”, “query to Book”, “list Category-wise” and “Transaction” navigation classes are automatically generated by the Navigation design model using the proposed WAD tool.
4.3 Presentation Transformation Module and the EA-OOH Presentation Model (PM)

The Presentation module is the third and final step towards the transformation from the requirements to the design model i.e., it involves the transformation of the complete Presentation WebGRL diagrams into the EA-OOH Presentation design model.

![Figure 9. The Screenshot of the Output EA-OOH Navigation Design Model for the Online Bookstore](image1)

The Presentation WebGRL consists of presentation goals, soft goals, presentation links, and associations between these goals. These presentation goals are used to represent the style and layout of the content on the web pages, frames, and windows at multiple levels depending on navigation links.

![Figure 10. The Screenshot of Input of Presentation WebGRL for Online Bookstore](image2)

The Presentation WebGRL diagram involving the location, appearance, and supplementary graphical components of the interfaces is used as input to obtain the EA-OOH Presentation Design model. To transform the WebGRL Presentation model, the presentation classes, as well as the navigation links...
and the content of domain classes, are used for designing the Presentation design model. The Presentation classes are identified and their associations are determined using navigation links for navigating to the next level of the web pages. The screenshot of the Input for Presentation WebGRL SubMenu for the proposed WAD Tool is shown in Figure 10.

The Transformation Algorithm 3 for transforming from the Presentation WebGRL diagram to the Presentation Design Model is presented in Figure 11. The Input to this algorithm is the Presentation WebGRL diagram to which the transformation algorithm steps are applied to generate the EA-OOH presentation model as an output. This transformation algorithm identifies the concepts of the presentation WebGRL model to be transformed into the presentation nodes and presentation links of the EA-OOH Presentation Model. Further, it specifies the components required to present level one and level zero of the Design Presentation Diagram.

The pseudo-code and the terminology used for transforming the Presentation WebGRL diagrams to EA-OOH Presentation model is as follows:

- A is a parent goal with a collection of links
- Xi are sub-goals i=1 to n
- Si are soft goals i=1 to m
- Ri are resource i=1 to p with resource name R
- AX(Li) are the navigation links attached from A to X for i=1 to q
- P is the main presentation goal used to represent an assortment of links for i=1 to r
- Pi is the presentation goal for i=1 to n
- PPi is the page chunk or presentation page for a presentation goal Pi for i=1 to m which may be displayed using a window or a frameset.
- Mi are Means-end tasks attached to a sub-goal or a parent goal which will be presented as Interface Components, include cell or include a frame of that Presentation Page for the parent or sub-goal
- PCI are the presentation classes associated with a frameset to display content using Interface Components for each navigation class for i=1 to n.
- ICI are the Interface Components to represent the content on a frameset or frame using image, text, audio, video, anchor, or a form.
- IFi are the Include cell or include frame components for i=1 to m.
- AX (Ai) represents the Interface Components anchor or a collection of anchors for i=1 to r.
- DCi (Ri) are the domain classes for goals attached to a resource R for i=1 to p.

<table>
<thead>
<tr>
<th>Algorithm 3 Algorithm for Transformation to EA-OOH Presentation Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Presentation WebGRL Diagram</td>
</tr>
<tr>
<td><strong>Output:</strong> EA-OOH Presentation Design Model</td>
</tr>
<tr>
<td><strong>For</strong> the main parent goal A, Create a homepage with a collection of anchors i.e. Main or Homepage Pi with associated IC frameset; // To represent content and anchors to navigate to other presentations goals now represented by Presentation Pages; //</td>
</tr>
<tr>
<td><strong>Do</strong> for each level till done for levels i=1 to n</td>
</tr>
<tr>
<td>Create PPi with ICI associated to a PCI for each corresponding NCi in the Navigation Model; // Each Presentation Page represents the content on a frameset with one or more collections of anchors; //</td>
</tr>
<tr>
<td><strong>For</strong> each navigation link from Pi to its subgoals check the type of link // To identify the association type between these Presentation Pages //</td>
</tr>
<tr>
<td><strong>Case 1:</strong> (AX(Li) is a simple navigation link) Then (AX(Li)= AX(Ai) there exists an anchor IC between Ai and Xi and create a PPi corresponding to Xi if it does not exist; //where Ai is the parent Presentation Class Page and Xi is the anchor associated Presentation Class Page //</td>
</tr>
<tr>
<td><strong>Case 2:</strong> (AX(Li) is a simple contribution link between A and Xi ) Then (AX(Li)= AX(Ai) between Ai and Xi where Ai is the parent goal and Xi is the</td>
</tr>
</tbody>
</table>
contributing presentation goal. create Ai and Xi= PPi associated to a PCI with ICi}// to represent the content on a frameset or frame using an image, text, audio, video, anchor, or a form.//

Case 3: (AX(Li) is a decomposition link between A and Xi)
Then {AX(Li)= AX(Ai) between Ai and Xi where Ai is the parent goal and Xi is the sub-presentation goal. Create Ai and Xi= PPi associated to a PCI with IC a collection of anchors to all its decomposed PPi with ICi }//to represent the content on a frameset or frame using image, text, audio, video, anchor, or a form //

Case 4: (AX(Li) between Ai and X=Mi where Mi is some Means-end task) Then (Mi= IFi or ICi) //In case it's a Means-end task then it is a frameset or an Interface Component. //

Endcase
Endfor
Enddo
Endfor

Figure 11. The Presentation WebGRL to EA-OOH Presentation Model Transformation Algorithm

The “Home Page” of the Online Book Store with “Presentational Classes, Frames, interfaces, Components, and FrameSets” represents the main presentation goal of the Online Bookstore Software System as shown in Figure 12.

Figure 12: Presentation Web-page of the Online Book Store

The Main Menu links the anchored collection interface components to other “Presentation Goals”. The next Presentation goal of the WebGRL i.e., “allow smooth searching options” is offered by the contribution links to presentation goals viz. ”query to search the book using the title”, ”query to search the book using the author” and ”List books category-wise”. All the presentation goals are represented by the navigation from the Main Home Page menu using various anchors viz. ”query to search the book using the title”, ”query to search the book using author” and ”List books category-wise” etc. to their respective presentation pages.

The final output of this transformation results in the EA-OOH presentation design model with the various presentation web pages that will be generated in the Online Bookstore as shown in Figure 13. In the center, the Online Bookstore Main Web page is presented and on the sides are the “Author”, “Category”, “Title”, and the “Transaction” web pages to which users can navigate from the main web page.
The proposed WAD tool assists the Design Engineer in creating a detailed web-oriented design model using a structured process of software development where both the requirements and the design phase have been dealt with in detail. Further, the design output models are UML Compliant that helps in adapting the output easily into a UML meta-model (Srivastava, 2015), thereby reducing the effort of the software engineers.

5. CONCLUSIONS

The requirements for web applications are different from traditional software applications due to the varying nature of the stakeholders and the development platforms involved. To cater to the diversified stakeholders, an automated web application development tool is presented that caters to the needs of the design engineers for web applications. The main goal of the design process is to deliver solutions that are easy to implement and maintain. The design solutions of a system form the pivotal elements influencing the maintainability of a system. A simple and understandable design will go far in simplifying the work of the designer as well as the maintainer. Also, the designer has to perceive the trade-offs and accomplish the best sense of balance among the various alternatives for design solutions. The proposed automated WAD tool primarily targets the requirements of web applications and their transformation to a design model. The WAD tool takes the WebGRL diagrams as input through the input forms and outputs the three design models that are refined, and can be evaluated for better alternative solutions. Besides this, the output design models generated are UML Compliant, thereby further reducing the workload of the software engineers.

The proposed transformation tool was also used by academicians and developers and the results have been overwhelming as it shortens the development cycle time for the web applications. Also, alternate designs can be generated in a short period. The evaluation of the design alternatives assists to choose the best design further improving design quality. The WAD tool completely automates the design phase of web applications using a well-structured, systematic, and automated transformation process.

REFERENCES

Authors Profiles

Prof. Sangeeta Srivastava received Post-Doctorate in Computer Science from UPES, Dehradun and Ph. D. degree in Computer Engineering from Faculty of Technology from Delhi University, Delhi, India. She is a Professor in Department of Computer Science, Bhaskaracharya College of Applied Sciences. She has published more than 30 papers in international journals and conferences Her research interests include Requirements engineering, software engineering, Design Engineering and machine learning.

http://www.webology.org
**Dr. Anuja Soni** is an Associate Professor in Deen Dayal Upadhyaya College, University of Delhi and has about 21 years of teaching experience. She has done Ph.D. from University of Delhi, Delhi, India. She has published various papers in international journals and conferences. Her research interests include Software Engineering, Fuzzy Logic, Multi-Agent System, Soft Computing, and Machine Learning.

**Prof. Vibha Gaur** received Ph.D. Degree in Computer Science from Department of Computer Science, University of Delhi, Delhi, India. She is a Professor in Department of Computer Science, Acharya Narendra Dev College. She has published more than 40 papers in international journals and conferences. Her research interests include Requirement Engineering, Software Quality, Fuzzy Logic, etc. She is a corresponding author for the article.