Received April 21, 2019, accepted May 8, 2019, date of publication May 20, 2019, date of current version June 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2917674

Automated Model-Based Test Case Generation for Web User Interfaces (WUI) From Interaction Flow Modeling Language (IFML) Models

NAZISH YOUSAF[1], FAROOQUE AZAM[1], WASI HAIDER BUTT[1], MUHAMMAD WASEEM ANWAR[1], AND MUHAMMAD RASHID[2]

[1]Department of Computer and Software Engineering, College of Electrical and Mechanical Engineering, National University of Sciences and Technology, Islamabad 46000, Pakistan
[2]Computer Engineering Department, Umm Al-Qura University, Mecca 21421, Saudi Arabia

Corresponding author: Muhammad Waseem Anwar (waseemanwar@ceme.nust.edu.pk)

ABSTRACT Since the emergence of web 2.0, the architecture of web applications has been transformed significantly and its complexity has grown enormously. In such web applications, the user interface (UI) is an important ingredient and with the increased complexity, its testing is getting increasingly complex and cost/time-consuming process. Recently introduced, interaction flow modeling language (IFML) is an object management group (OMG) standard. IFML is gaining popularity for developing web applications, primarily, because of its excellent features for modeling UI elements, their content, and their interaction capturing capabilities. However, despite its superior UI modeling features, its UI testing is accomplished through traditional time-consuming techniques, which are employed after implementing the UI code. Hence, to overcome these limitations, this paper introduces a novel model-based testing approach for IFML UI elements. The proposed approach provides complete navigation testing using formal models. Moreover, the approach transforms the IFML models to all necessary UI testing artifacts by generating state transition matrix plus detailed UI test case document. As a part of a research, model-based user interface test case (MBUITC) generator tool is implemented to automatically generate navigation model for formal verification, test case document, and transition matrices from IFML models. The applicability of the proposed approach is validated through two benchmark case studies. The results have shown that the proposed approach provides test cases at the early stages of development, i.e., specification and analysis, which eventually helps in building a right product at the right time at a comparatively lower cost.

INDEX TERMS Formal verification, IFML, MBT, model-based testing, UI, web applications, WUI.

I. INTRODUCTION

World Wide Web (WWW) is an information space which contains data, documents and web resources all over the world connected by webpages. It has become an ever-increasing network of interlinked pages and web applications, packed with photos, videos and other interactive contents, therefore there is an increased complexity in the architecture of web-based applications [1]. Over the last decade, the usage of WWW has increased significantly due to the involvement of social media (e.g. Facebook etc.) and people access web apps through various devices i.e. laptops, PDAs, desktops and mobile phones. Consequently, the primary goal of modern websites is the consistent representation of information, so that, the contents can be accessed / shared on different devices. Furthermore, it is also important to provide universal interfaces in order to accommodate the demands of users from different countries. As a result, the efforts and time require to develop and test Web User Interfaces (WUI) are significantly increased [2]. There is a strong need to minimize the development complexity of WUI to achieve certain goals like usability and productivity for modern websites. This can be achieved by applying the latest Model Driven Engineering (MDE) techniques [3]–[5].

Model Driven Engineering (MDE) is a well-known methodology of software engineering which is commonly

The associate editor coordinating the review of this manuscript and approving it for publication was Bora Ona.

applied in different domains like embedded systems [3], information systems etc. to simplify the development process and provide early design verification features. Furthermore, it is fully applicable for the development of both web and desktop software applications [6]. Therefore, it is frequently applied to simplify the development of modern web applications. For example, UML (Unified Modeling Language) has been widely used as a standard General Purpose Modeling Language (GPML) for the visualization of design and functionality of different web applications. However, UML lacks in covering the significant characteristics of web applications i.e. user interaction and interface. Therefore, WebML (Web Modeling Language) with its own notations was introduced for the development of complex web applications. In 2013, OMG (Object Management Group) evolved WebML into IFML (Interaction Flow Modeling Language) [7]. IFML is designed to capture the structure, user interaction and control flow of front-end of web applications. Furthermore, it provides support for platform independent level description for the UI of web applications accessed on any kind of device i.e. desktop, computer, laptop, PDA, mobile or tablet independent of the residing implementation techniques or platforms. In order to provide platform independent level interaction, IFML provides a stable set of concepts used to capture the fundamentals of user interaction with the interface of web applications.

To deal with the complexity of modern web applications and particularly user interfaces, the development and testing has to start right from the requirement gathering phase. In this context, IFML facilitates the design process by building the design models of the web application including the user interaction, content and structure of application. However, IFML does not provide design verification and testing capabilities so far. Consequently, the design of web application is developed through IFML and separate approach in late development phase is introduced to test the functionality and user interfaces of web application. This significantly affect the achievement of important factors like quality, efficiency and Time to Market (TTM) [8]. Therefore, there is a strong need to develop an approach to automatically generate test cases from IFML models. This leads to perform the design verification of user interfaces in initial development stages to make the testing process more precise and efficient. Furthermore, it also ensures the cross-platform compatibility of user interfaces as IFML models are developed at platform independent level.

This article presents a novel approach to automatically generate WUI test cases from IFML models. Particularly, IFML models containing web application functional and user interfaces requirements are transformed into test cases, exercising particular actions in the application to verify its expected behavior. The contributions of the article, as shown in **Fig. 1** are summarized as follows:

- A novel methodology is proposed to automatically generate test case document, state transition matrix, source and target information matrix and timed automata navigation models from the initial collected web application requirements which are represented through domain (UML) and front end (IFML) models.
- Firstly, the conceptual mapping between IFML constructs and web application concepts is performed (**Section III-A**). This mapping provides the platform for auto generation of WUI test cases from IFML models.
- Secondly, four types of transformation rules are developed (**Section III-B**) to generate target test case artifacts from the IFML models.
- Thirdly, a complete Model-based User Interface Test Case (MBUITC) generator tool is implemented in JAVA and Acceleo (**Section IV**), capable of generating accurate target test case documents from the IFML models.
- Finally, the validation (**Section V**) is performed through two bench mark case studies i.e. Online Auctions and Library system case studies. Navigation verification is performed through UPPAAL tool by utilizing auto generated navigation model. Two properties have been considered for the verification of pages navigation i.e. Reachability and Deadlock.
- The comparative analysis of proposed approach with state-of-the-art is given in **Section VI**.

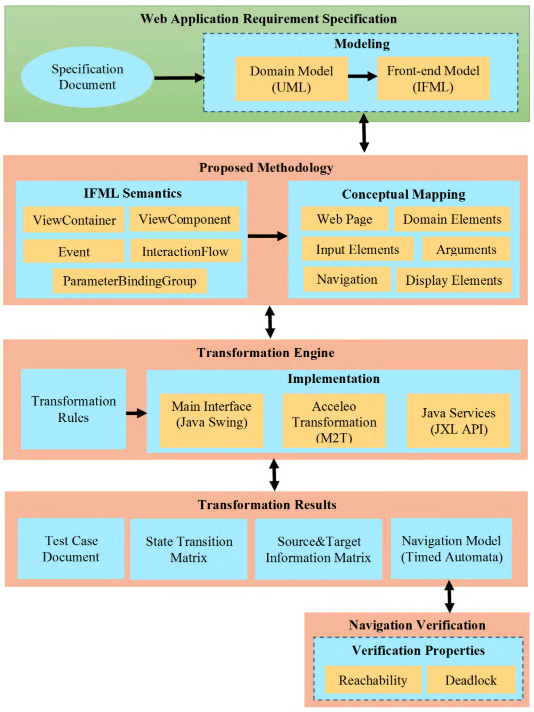


FIGURE 1. Overview of the research.

Most important contribution made by the proposed approach is automated navigation verification. IFML model is transformed into a navigation model which contains the view containers as states and navigation between them as transitions. The resulting Timed Automata model can be used to verify the reachability and deadlock properties; hence, providing fully automated navigation verification.

Other contributions of the research are automatic generation of State Transition matrix, Source & Target Information Matric and UI Test Case Document. The generated matrices are used as a black box testing technique and are useful for quick state transition testing of the web application. They help testers to expose the invalid or unintended states and contain test case for each navigation in the IFML model whereas the UI Test Case document generated is a complete testing artifact which provides full coverage of all the test cases used for UI testing of a web application.

II. PRELIMINARIES

In this section, we briefly discuss the ground works for this research. Introduction of IFML is provided in **Section II-A**, and literature review is discussed in **Section II-B**.

A. INTERACTION FLOW MODELING LANGUAGE (IFML) STANDARDS

The Interaction Flow Modeling Language (IFML) was standardized by OMG in March 2013. It was inspired by WebML and WebRatio experience which were used for model driven web-based application development. It has widely been adopted since then. IFML itself describes how we can apply model driven engineering (MDE) to the problem of front-end design of software applications. IFML is designed to capture the structure, user interaction and control flow of front-end of any software application. Furthermore, it provides support for platform independent level description for the GUI of any software application accessed on any kind of device i.e. desktop, computer, laptop, PDA, mobile or tablet. In order to provide platform independent level interaction, IFML provides a stable set of concepts used to capture the fundamentals of user interaction with the interface of software applications, defined in subsequent section.

1) DOMAIN MODELING

Requirement specification contains the textual information on what should be the structure of the application or what functions should be performed by it. It provides the user roles, domain entities and the relationship between roles and use cases. Domain modeling is referred as a highly relevant and complementary activity to front-end modeling. In order to design an IFML model, UML model containing the domain concepts of the application is required. This UML model is simply a UML class diagram contains information about the objects identified in requirement specification phase. The resulting model encompasses classes, attributes and relationships between classes that are later used in the IFML model which is used to map the domain concepts provided by UML domain model to the front-end of the application.

2) IFML MODELING

IFML provides support for the front-end application specifications without taking in account the underlying technological details. IFML provides support for the visualization units through which interface is composed, content to be displayed, events and actions involved, their effect on the interface state and the parameters to be passed while the units communicate. In short, IFML sums up all these concepts in one diagram unlike UML which relies on multiple diagrams to express each concept.

B. LITERATURE REVIEW

Literature review is performed in this section to highlight the applicability and usability of IFML in the field of software application development. M. Brambilla et al. [9] propose a multi-step approach for both mobile and web development. The study present how model driven transformation can ease code generation from requirement specification. The approach started from requirements specification and business specifications and specified them in IFML based platform independent generic model that can be used for both web and mobile aspects. Later, this model was transformed into a more domain specific model for mobile by blending the IFML model with a mobile specific modeling language called MobML. Code generation is done for both web and mobile specific platforms.

C. Bernaschina *et al.* [10] propose MDE based approach for improvement of application development process by combining visual application modeling language (IFML) with web log analytics. By using this approach, an appropriate blend can be achieved which helped in user behavior analysis. R. Acerbis *et al.* [11], [12] highlight the usage of mobile extensions of IFML which can be used for rapid application development of mobile applications. A comprehensive toolset called WebRatio is proposed for this purpose. WebRatio supports IFML and its web extensions and has some model checking and code generation features resulting in generation of code for cross-platform mobile applications using Cordova framework.

S. Roubi *et al.* [5], [13] adopt IFML as a modeling language for Rich Internet Applications (RIAs). The paper proposes an approach for Model-based Graphical User Interface (GUI) generation for RIAs by transforming the IFML models using Eclipse tool. Each input element of IFML model is mapped to an element of RIA meta-model in order to automatically generate code for a running application. Major focus of this approach is on GUI elements which are considered very important for any web-based application.

Brambilla *et al.* [14] present their work on model-driven approach extending IFML for the development of cross-platform mobile applications. The proposed extensions cover both the user interface and event aspects of components. Screen has been introduced to fill the part of ViewContainer in a mobile application. Similarly, ToolBar has been introduced as a sub-container. MobileSystem stereotype is introduced for mobile ViewContainer and MobileComponent stereotype is introduced for mobile ViewComponent in IFML standard. Similarly, several events have also been introduced in the context of mobile application. The study highlights that IFML is strong enough to capture all the aspects of a typical mobile application.

Nieto *et al.* [15] present a model driven approach for development of a web mobile application named ocioColombia. In this paper, IFML is used for development of this application in order to generalize it from web domain to general domain. Umuhoza and Brambilla [16] carry out an extensive survey for identification of different model driven approaches used for the development of mobile applications. The survey provides a comparative analysis of research approaches and the commercial solutions identified on the basis of SDLC phases, application aspects, Model Driven Development (MDD) techniques, application types and supported platforms.

Rhazali *et al.* [17] propose a model transformation approach in order to transform Computation Independent Model (CIM) to a web-based Platform Independent Model (PIM) according to Model Driven Architecture (MDA). To provide web view to PIM model, SoaML and IFML has been used. A comprehensive set of rules are constructed in order to develop UML activity, class, use case and state diagrams at CIM level. Then, a set of transformation rules have been defined for transformation from CIM to PIM i.e. SoaML and IFML models. This approach provides a solution to the problem of conversion of business models at CIM level to design and analysis models at PIM level.

Laaz and Mbarki [6], [18] use MDE methodology to generate user interface from abstract models in order to meet requirements for RIAs. The approach combines ontology with IFML. Ontology domain captured the UI logical description and IFML has been used to capture the interaction between domain elements. Abstract model is given as input and two transformations are applied on it. First transformation converts the PIM to Platform Specific Model (PSM) and then through second transformation on PSM, flex interface code for RIA is generated.

Fraitak *et al.* [7], [8] propose a testing technique which used IFML in order to generate the automated front-end tests. IFML is used to capture the interface components details. MDE is used for transformation of IFML model to abstract front-end test model for the System under Test (SUT). Some rules are defined according to which the front-end model is transformed into executable test cases. The abstract test case scenarios are transformed into physical test case scenarios using template engine. Generated test cases can be executed using Selenium. Umuhoza *et al.* [19] propose a strategy for WebRatio in developing a MDD tool for mobile application development. The approach uses PIM level language IFML's extension for mobile application development. The extended language Mobile IFML supports design views representing the structure, design and interaction flow between mobile interfaces. The study also highlights some code generation approaches for mobile applications. In another work Salini *et al.* [20] present an approach in which web analytics are used to automatically generate navigation models for mobile applications. The proposed approach uses usage patterns from both mobile and web usage and provides an automated transformation into mobile based navigation tree and design models.

Bernaschina *et al.* [21]–[23] present an open source tool for rapid prototyping of web and mobile applications using MDD approach. The approach applied model to model and model to text transformations. Model to model transformation on IFML diagram result in place chart nets which later helped in simulation and verification. Model to text transformation, on the other hand resulted in code generation for web and mobile domain form IFML model.

Anwar *et al.* [3], Larissa da Costa *et al.* [28] introduce Model-based User Interface (MBUI) approach used for user interface engineering for the domain of web-based systems. The UI stereotype is used to capture UI specification, behavior and its presentation. This concept has been applied along-with IFML for describing a web portal user interface stereotype as an interaction pattern for automatic web portal component generation. Bernaschina *et al.* [30] in a research paper propose formalization of IFML semantics by mapping IFML constructs to equivalent concepts in Place Chart Nets (PCN) in a model driven environment for generation of code for all web and mobile applications. Rodriguez-Echeverria *et al.* [31] propose a MDWE tool called AutoCRUD which is a webRatio plugin and generates CRUD specifications in IFML automatically in order to increase the productivity of development teams.

III. PROPOSED METHODLOGY

A. TARGETED IFML CONSTRUCTS

IFML meta-model provides the semantics and structure of constructs used in IFML. UML profile in IFML meta-model defines the syntax used to express IFML models in UML. IFML meta-model comes with two packages. Core package contains main IFML concepts whereas the Extensions package contains some enhanced characteristics that make the application more interactive. The basic purpose of introducing extensions is to make application more expressive, increase the readability and to make the elements less abstract. This package majorly contained web, desktop, component and multi-screen extensions. Because of limited time and scope of the research, we have only targeted the core concepts and some of main extensions concepts which are mandatory for development of a Web User Interface. A brief description of the targeted IFML concepts is given as follows,

- <<**ViewContainer**>> IFML model consist of one or more view containers which basically are used to express web pages and windows in case of web applications and desktop applications respectively. View containers can be nested. Child containers can be displayed at the same time as of parent containers or they can be made mutually exclusive by using XOR nesting. In case of mutual exclusion, one container can be set as default, when user accessed parent container, default child container is also displayed. For Example, **Fig. 2** shows a simple IFML model from movies case study. It explains simple scenario that if the user wants to add a

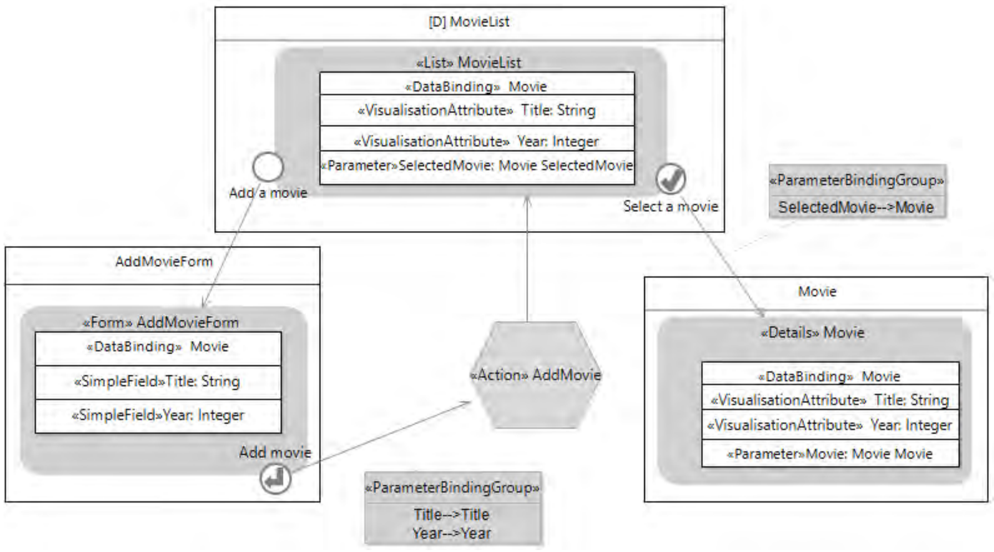


FIGURE 2. Example of IFML model.

new movie, form will be displayed, and user will add the asked input and add action will be performed resulting in saving the new data in MovieList. And if the user wants to see detailed information of movies data, the selected movie in MovieList will be displayed in detail. In given model, AddMovieForm and Movie are ViewContainers and MovieList represents a ViewContainer that has been set to default.

- <<**ViewComponent**>> In IFML model, a ViewContainer can contain one or many ViewComponents. ViewComponents contain the type of data to be displayed i.e. Form, List or Details which are included in extensions package of IFML metamodel. Input and output parameters can be associated with ViewComponents. Another important IFML concept is DataBinding which is part of ViewComponent. It represents the relation between the domain model element and ViewComponent. A DataBinding references a domain concept and DataBinding elements specify exactly what data needs to be extracted from the domain. **Fig. 2** shows the notation used for ViewComponents with specific extension type i.e. MovieList is used to represent movie data in the form of list. AddMovieForm represents ViewComponent used to take input data in a form whereas Movie is a ViewComponent type to display detailed information about selected object i.e. SelectedMovie.
- <<**Event**>> Events are used to express interaction between ViewContainers and ViewComponents. It causes a transition between source and target web page. There are many types of Events i.e. OnSubmit, OnLoad and viewElementEvent etc. In **Fig. 2**, Add a movie attached to the MovieList ViewComponent is a representation of viewElementEvent and Select a movie is a representation of OnSubmit event.
- <<**InteractionFlow**>> An InteractionFlow represents the effect of an event used to connect ViewComponents and ViewContainers. It characterizes the change of state of interface. Interaction flows in IFML are of two types i.e. data flow and navigation flow. Data flow represents the transfer of data between two IFML elements represented by dotted line and are not caused by user interaction whereas, navigation flow expresses the navigation between components and containers represented by solid lines as shown in **Fig. 2**.

B. PROPOSED SOLUTION

The proposed approach works on the idea of conceptual mapping of the targeted IFML constructs with the testing artifact elements which maps the concepts of IFML over the web testing elements in order to develop an understanding and to provide ease for the transformation process. In order to obtain the complete test case document for the web application testing, a M2T transformation is applied on UML and IFML model. The transformation rules are explained in the subsequent sections.

1) IFML TO TEST CASE TRANSFORMATION RULES

Test case document is a main artifact produced for testing process in order to check expected behavior of the application. Writing effective test cases and maintaining consistent format for them is rather time taking process. The proposed solution provides a complete test case document after the transformation. Mapping rules used for transformation of IFML model components into their respective test cases are provided in this section. *ViewComponent* have three extension types i.e. *Form*, *List* and *Details*. The transformation rules defined in **Table 1** are used to transform the UI components from IFML model to test cases. We have not included other components of IFML model for transformation because they are unable to capture the UI details.

TABLE 1. Transformation rules for IFML to test cases.

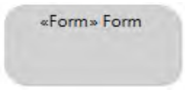
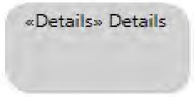
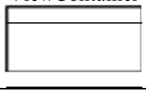
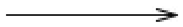
IFML Model	Test Case Document	Description
IFML Form 	Test case containing checks for form	*ViewComponent--Form*--name→Test case name *ViewComponent--Form--SimpleField*→input value *ViewComponent--Form--SelectionField*→input value *ViewComponent--Form--onSubmit*→final step (submit).
IFML List	Test case containing checks for list	*ViewComponent—List*--name→Test case name *ViewComponent--List--DataBinding*→Domain model element *ViewComponent--List--VisualizationAttribute*→Domain attributes to be displayed.
IFML Details 	Test case containing checks for details	*ViewComponent—Details*--name→Test case name *ViewComponent—Details--DataBinding*→Domain model element *ViewComponent--Details--VisualizationAttribute*→Domain attributes to be displayed.

TABLE 2. Transformation rules for IFML to state transition matrix.

IFML Model	State Transition Matrix	Description
ViewContainer 	State	*ViewContainer*--name→*First* row and column
NavigationFlow 	Transition	*NavigationFlow*→"T" and "F" values.

Actually, IFML ViewComponent alone is only used to represent the divisions inside a web page. Consequently, ViewComponent doesn't belongs to any meaningful testing concept in the given context. Therefore, it is not required to transform the viewComponent in MBUITC. However, the elements inside the *viewComponent* like *Form* and *List* etc. are relevant to given testing approach. IFML *ViewComponent* of type *Form* is mapped to its respective test case in the test case document. Name of *Form* is mapped to test case name. *SimpleField* and *SelectionField* of each form is mapped to check for the type of input value and selected value respectively. on *Submit* event on the form is mapped to final submit step in test case. IFML *ViewComponent* of type *List* is mapped to its respective test case in the test case document. Name of *List* is mapped to test case name. *DataBinding* in the *List* is mapped to domain model element and *VisualizationAttribute* is mapped to the domain element attributes to be displayed in the list. IFML *ViewComponent* of type *Details* is mapped to its respective test case in the test case document. Name of *Details* is mapped to test case name. *DataBinding* in the *Details* is mapped to domain model element and *VisualizationAttribute* is mapped to the domain element attributes to be displayed in detail.

2) IFML TO STATE TRANSITION MATRIX TRANSFORMATION RULES

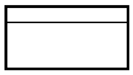
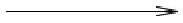
State transition matrix contains true and false values for transitions from one state to another. This matrix can be very helpful for quick black box testing or navigation testing. The navigation model is generated based on this matrix. In this section, we explain the mapping rules used for our transformation of IFML core model to the State Transition Matrix. The transformation rules defined in **Table 2** are used to transform the concepts from IFML model to the State Transition Matrix.

ViewComponent in IFML core model is transformed into state, in the State Transition matrix. Each *ViewContainer* name is mapped to elements of first row and column of the State Transition matrix. *NavigationFlow* in IFML model is mapped to transition in the State Transition matrix where it is represented as "T" (true) and if the transition does not exist, it is represented as "F" (false).

3) IFML TO SOURCE & TARGET INFORMATION MATRIX TRANSFORMATION RULES

Source&Target Information Matrix includes details of source and target pages and the parameters passed during the navigation between pages. Each row of Source&Target Information Matrix contains test case for each navigation in the IFML model. It assesses the complete detailed functionality of web-app like the flow of information across it. Web application must be tested against the irregularities caused due to poor handling of allowed data type and parameters or arguments passed between source and target pages. In this section, we explain the mapping rules used for our transformation of IFML core model to the Source&Target Information Matrix

TABLE 3. Transformation rules for IFML to Source& target information matrix.

IFML Model	State Transition Matrix	Description
ViewContainer 	Source/Target name	*ViewContainer*→Source and target states. *ViewContainer*--(isLandmark=true)→true value *ViewContainer*--(isDefault=true)→true value *ViewContainer*--(isXOR=true)→true value
NavigationFlow 	Navigation	NavigationFlow→Navigation
ParameterBindingGroup	Parameter	*ParameterBindingGroup--ParameterBinding*→parameter *value*

containing the detailed test cases for each navigation in the IFML model. The transformation rules defined in **Table 3** are used to transform the concepts from IFML model to the Source&Target Information Matrix.

ViewContainer in IFML core model is mapped with source and target in the Source&Target Information Matrix. One container can act as source in one case and target in another. "isLandmark", "isDefault" and "isXOR" attributes of each *ViewContainer* as "true" and "false" values are mapped to their respective cells in the matrix against the name of *ViewContainer*. NavigationFlow in IFML model is mapped to navigation. Only in case of navigation, rows are added in the Source&Target Information Matrix. Each *ParameterBinding* inside the *ParameterBindingGroup* is mapped to name of parameter passed during navigation in the Source&Target Information Matrix where it can contain a value or can be null if there is no *ParameterBinding*.

4) IFML TO TIMED AUTOMATA TRANSFORMATION RULES

Navigation testing of a web application is of great importance in order to analyze the application architecture and to improve critical user flow. A navigation model is best way to depict navigation visually and can be used for deadlock and reachability verification. Mapping rules used for the transformation of IFML core model to timed automata are provided in this section. These rules result in an equivalent navigation model for verification purposes. We have used Eclipse IFML plugin for modelling of IFML core model and UPPAAL model checker has been used for verification of resulting Timed Automata navigation model. UPPAAL provides simulation and verification of the models in which timed automata has been used. Timed Automata model is comprised of states and transitions [24], [25]. *Location* in Timed Automata model represents a state and *intiallocation* is used to represent initial state of the model. *Edge* in Timed Automata model is used to represent transition. For our approach, it is not essential to include the synchronizations or time and guard constraints in Timed Automata model because here we are dealing with simple navigation testing only, therefore, it is not required to use synchronization concept. Only a navigation model is required for verification of reachability and deadlock in our web navigation, so only states, transitions and navigations have been used. *ViewContainer* in IFML model typically is used to represent page in a web application and screen in a mobile application. Transformation rules for view containers and navigation flow are provided in **Table 4** along-with the graphical notations of source and target transformation.

ViewContainer mentioned as Home and containing value *"true"* for *isDefault* attribute, representing home page in the IFML model is mapped to the *initiallocation* in the Timed Automata model. *Initiallocation* is used to represent the initial state of the system. *NavigationFlow* in the IFML model is mapped to *Edge* in the Timed Automata model. In order to create an Edge between two locations in Timed Automata model, at least one Navigation Flow should exist either between two view containers, view components of two view containers, any view component of one view container and other view container or vice versa. Both of these notations represent transition from one state to another. *ViewContainer* representing a page in the IFML model is transformed into *location* in the Timed Automata model where it represents state. ViewComponent in a ViewContainer is not transformed and containing ViewContainer is taken as a state.

We have not included other components of IFML model for transformation and hence their rules are not included because they did not lie under our area of focus. Details of applicability of these rules is described in **Section V**. Two case studies have been used to verify the transformation rules mentioned above.

IV. MODEL-BASED UI TEST CASE GENERATOR (MBUITC)

As a part of research, **M**odel-based **UI** **T**est **C**ase (MBUITC) Generator tool is developed as per transformation rules (**Section III**). Firstly, it provides facility to model IFML model in Eclipse IFML editor. Secondly, it provides a transformation engine that transforms the IFML model and provides testing artifacts. Finally, it provides the facility of model verification using Timed Automata formalism.

Acceleo is an Object Management Group (OMG) standard tool that provides sophisticated features to perform Model to Text (M2T) transformations. It is frequently used in Model Driven Engineering to achieve both simple as well as complex transformations. Here, we use Acceleo in MBUITC to implement a transformation engine that fully automates the testing

TABLE 4. Transformation rules for IFML to timed automata model.

IFML Model	Timed Automata Model	Description
Home ViewContainer ▭	Initial location ◎	*ViewContainer*--Home(*isDefault=true)*→*initiallocation*
NavigationFlow →	Edge →	*NavigationFlow*→*Edge*
ViewContainer ▭	Location ●	*ViewContainer*→*location*

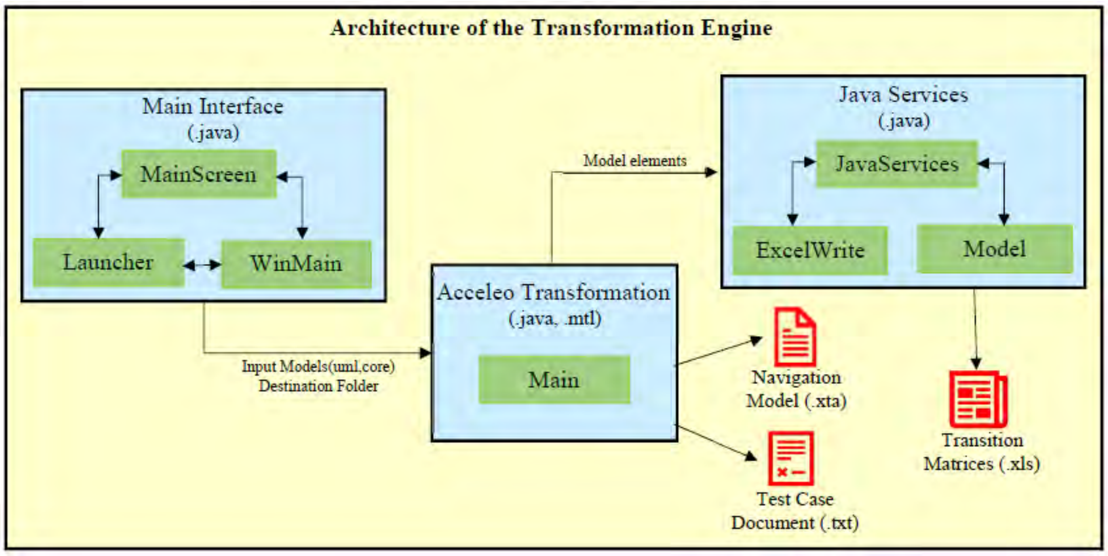


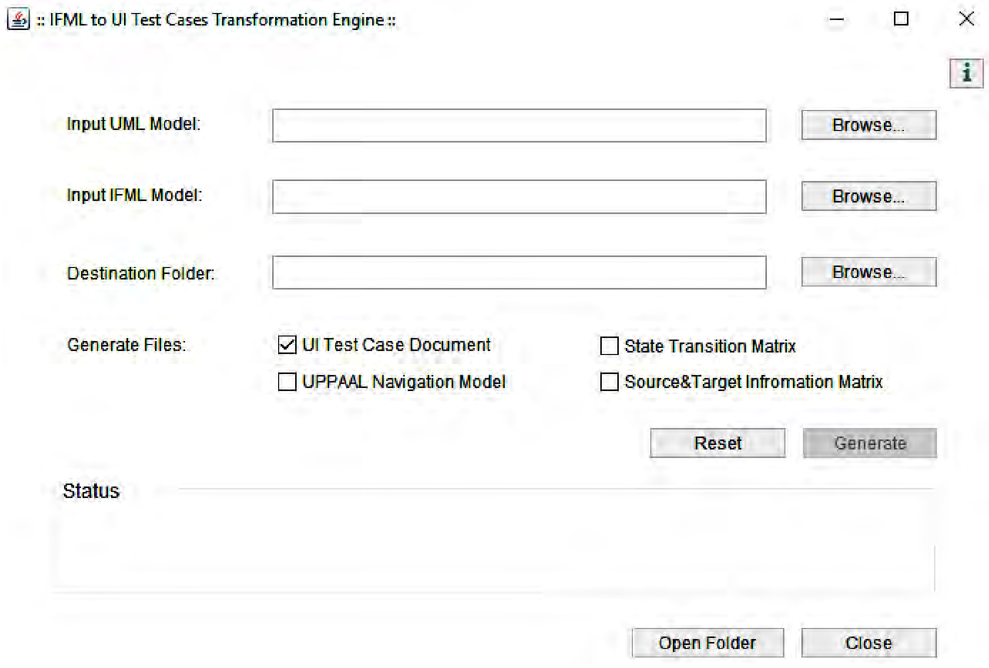
FIGURE 3. Transformation engine architecture.

phase of SDLC by providing automated test cases from IFML models. JAVA language and Acceleo tool is utilized for the implementation.

MBUITC transformation engine is composed of three main components which are Main Interface, Acceleo Transformation and Java Services as shown in **Fig. 3.** ***Main Interface:*** Main Interface component consists of three classes i.e. *MainScreen.java, Launcher.java* and *WinMain.java* which are used in development of graphical user interface of MBUITC. MBUITC Generator tool provides three options i.e. IFML Editor, Transformation Engine and Navigation verification. We have selected UPPAAL model checker for navigation verification of the generated Timed Automata navigation model. IFML Editor and UPPAAL can be opened directly whereas the interface of our transformation engine is shown in **Fig. 4**. The transformation engine takes *UML* and *IFML* models as input using a *Browse* button. It also asks for path of *Destination folder*. Checkboxes are provided so that the user only generates the output files of choice. By clicking the *Generate* button, the engine generates the selected files. Along-with test case document (.txt), State Transition matrix and Source&Target Information matrix are generated in xls format. Another important artifact generated by transformation is Navigation model with xta extension which can then be opened in UPPAAL [25]. To show the status of transformation, a *Status* bar is provided. *Reset* button, empties all the fields i.e. input models path, destination folder path, status and all checkboxes except for the test case document checkbox which is by default checked. *Close* button closes the interface from the screen.

A. ACCELEO TRANSFORMATION

Main interface takes UML and IFML models as input and passes them to Acceleo Transformation. Acceleo transforms the input UML and IFML models in the respective testing artifacts. Foremost files included in Acceleo Transformation are *main.java* containing java code for transformation and *main.mtl* containing Acceleo transformation code. These two files work together to produce Test Case document (.txt) which contains all the UI testing related concepts captured from IFML model. The Test Case document covers test cases related to three main elements of IFML i.e. List, Form and Details explained earlier in **Section III-A**. We have implemented the transformation rules in main.mtl file through


:: IFML to UI Test Cases Transformation Engine ::

Input UML Model: Browse...

Input IFML Model: Browse...

Destination Folder: Browse...

Generate Files: [✓ UI Test Case Document] State Transition Matrix UPPAAL Navigation Model Source&Target Infromation Matrix

Reset Generate

Status

Open Folder Close

FIGURE 4. IFML to test cases transformation engine.

acceleo code. Therefore, main.mtl file is responsible for the generation of target testing artifacts from IFML source models. However, main.mtl file cannot be executed directly, and it requires initiation from java code. Consequently, it is initiated through main function of main.java file. Main.mtl then generates the output xta file containing code in Timed Automata formalism, describing the navigation model for the input IFML. This xta file has proper syntax and semantics, local declarations are written at first, then states and transitions are written. This file then can be opened in UPPAAL model checker tool to check the deadlock and reachability [29].

B. JAVA SERVICES

Java Services are developed using three main classes, JavaServices.java, ExcelWrite.java and Model.java. *Model.java* class only used to get the input IFML model and instantiates the IFML model. *JavaServices.java* is the main class used to store particular details regarding transition matrices because acceleo does not provide the straightforward facility for storing data through main.mtl file. Therefore, in order to store transition matrices details, we have implemented *JavaServices.java* class. The functions defined in *JavaService.java* class can then be used inside *main.mtl* file using queries. *ExcelWrite.java* is only used to shape our matrices in the form of excel sheets. One of the generated matrices contains only navigation related data and the other one contains complete information on all the paths in our IFML model.

After successful transformation, the transformation engine provides state transition matrix containing true and false values for transitions from one state to another. Secondly, it provides another Source&Target Information Matrix which includes details of name of source and target pages. Attributes like isLandmark, isDefault, isXOR of source page are given. Parameters or arguments passed between source and target pages are also provided in this matrix. Each row of Source&Target Information Matrix contains test case for each navigation in the IFML model. Actually, the state transition matrix and Source&Target information matrix provide the foundation for the proposed approach. In fact, these matrices are really significant in the area of Model Based Testing (MBT) and frequently used in different studies. For example, Suhag and Bhatia [1] propose a novel MBT approach for web applications where Source&Target Information matrix is developed to generate sequence diagram. In another study [2], dealing with Model-driven GUI testing, State Transition matrix is developed to generate navigation model and test oracles. Particularly, state transition matrix and Source&Target information matrix serve as intermediate formats that provide basis to generate test case document and timed automata navigation model. Thirdly, it gives a test case document containing detailed UI test cases and this document itself is a main testing artifact. Lastly, a navigation model is obtained which is used for deadlock and reachability verification. The complete source code of MBUITC Generator tool along with user manual and sample case studies can be found at [32].

V. VALIDATION

In this section, we present the applicability and validity of our proposed approach with the help of two case studies of different sizes. Details and validation of Online Auctions case study is given in **Section V-A** and Library case study is given in **Section V-B**.

A. ONLINE AUCTIONS CASE STUDY

We explain and validate Online Auctions case study using four sections. **Section V-A-1** covers the requirement specification for an Online Auctions website. **Section V-A-2** and **Section V-A-3** present the UML domain modeling and IFML modeling of this case study in Eclipse IFML editor, respectively. Lastly, the transformation and verification of the case study modeled in IFML has been provided in **Section V-A-4.**

1) REQUIREMENT SPECIFICATION

a: MASTER PAGE

The Online Auctions website should have a landmark page called Master page which should be accessible from all other pages. Master page should contain a section showing details of the User, it shows user name, first name, last name and score of the current logged in user. This section should contain a mouse over button which should lead the master page to Logged in user menu page. Master page should have another section showing the cart details i.e. number of items and logged in users. The user name data will be taken from user details section. Master page should also contain a Search form with an input field that takes String input for search key and a drop-down list showing all the available category names. This form should contain show suggestions and hide suggestions buttons. It should also have a Search button which should lead the page to Search results page. Another button named "Advanced" should be attached to this form which should lead the page to Advanced results page. Master page should have sign in, register and shop by category buttons. Sign in button should lead the page to Sign in page. Register button should lead the page to Register page and shop by category button should lead the page to Category tree page.

b: LOGGED IN USER MENU

Logged in user menu page should contain a section with a list of user details. Sign out, account settings and my collections buttons should be attached to this list. Logged in user menu page should be accessible from master page.

c: SIGN IN

Sign in page should contain a section with a Sign in form containing sign in credentials. Submit and register buttons should be attached to this form. Submit button leads the page to Home page and register button leads towards the Register page. Sign in page should be accessible from master page.

d: REGISTER

Register page should contain a section with a registration form containing all the necessary information with a Submit button which redirects towards Home page. Register page should be accessible from master page.

e: CATEGORY TREE

Category tree page should contain a section showing list of shop by category with their names with a Select button. Category tree page should have two buttons; all categories and trending collections. Category tree page should be accessible from master page.

f: CATEGORY

Category page should contain a section showing details of Category with their names. Category page should also contain a section showing list of features with their names, images and links with a button attached to it which takes feature link/address as parameter and leads the page towards Show feature page. Category page should contain a section showing list of listing groups with their names and images with a button attached to it which takes feature selected group as parameter and leads the page towards Groups page. Category page should contain a section showing the details of events according to names and has a button attached to the list which takes selected events as parameter and leads the page towards Event page. Category page should contain a section showing the details of Brands according to names and has a button attached to the list which takes selected brand as parameter and leads the page towards Brand page. Category page should contain a section which contains list of peer categories according to names and has a button attached to the list which takes selected category as parameter and redirects the page to itself sending the parameter to Category details section. Category page should contain a section which contains list of sub categories according to names and has a button attached to the list which takes selected category as parameter and redirects the page to itself sending the parameter to category details section. Category page should be accessible from Home and Category overview pages.

g: EVENT

Event page should contain a section having event details. Event page should be accessible from Category page and selected event should be passed as parameter.

h: BRAND

Brand page should contain a section having brand details. Brand page should be accessible from Category page and selected brand should be passed as parameter.

i: SHOW FEATURE

Show feature page should contain a section having feature details. Show feature page should be accessible from Category page and Home page.

j: GROUPS

Group page should contain a section having group details. Group page should be accessible from Category page and selected group should be passed as parameter.

k: CATEGORY OVERVIEW

Category overview page should contain a section showing a list of sub categories and a button which takes selected category as parameter and leads the page towards Category page. Category overview page should also contain a section showing a list of other sub categories and a button which takes selected category as parameter and leads the page towards Category page. Category overview page should also contain a section showing details of Category with image and sends top category as parameter to both other section s. Category overview page should be accessible from Home page.

l: COLLECTIONS

Collections page should contain a section showing a list of Collections containing name, payoff, main image, description blob, username and photo information. This list also contains two buttons attached to see collection and seller. Collections page should be accessible from All trending and Home pages.

m: HOME

Home page should contain a section showing list of main categories according to their titles and has show, on select and mouse over buttons attached to it. Show button leads the page towards top category page. On select button takes selected category as parameter and passes it to Category details in Category page. Mouse over button leads the page towards Category overview page. Home page should also contain a section showing list of features with images and has a button attached to it which leads the page towards Show feature page. Home page should also contain sections for top collection, promoted collections and trending collections. Home page should have four buttons; collections, my feeds, all trending collections and category overview. Collections button leads the page towards collection overview page. My feeds button leads the page to sign in form section inside Sign in page. All trending collections button leads the page towards All trending page and category overview button leads the page towards Category overview page by passing the top category parameter to category details section inside Category overview page. Home page should be accessible from Sign in and Register pages.

n: SEARCH_RESULTS

Search results page should contain a section showing list of formats with their values and count. A button should be attached to this list which takes the selected format as parameter and sends it to the listings section. Search results page should contain a section having a list of conditions and has a button attached to it which takes the parameter selected condition to the listings section. Search results page should contain a section showing price list with their maximum and minimum price. A button should be attached to this list which takes the selected maximum and minimum price as parameter and sends it to the listings section. Search results page should also contain a section having a list of locations and has a button attached to it which takes the parameter selected location to the listings section. Search results page should contain a section having a list of delivery types and has a button attached to it which takes the selected delivery type as parameter to the listings section. Search results page should contain a section having a list of options and has a button attached to it which takes selected option as parameter to the listings section. Search results page should contain a section showing categories list and has a button attached to this list which takes the selected category as parameter and sends it to the listings section. Search results page should contain a section showing listings list with their title, price, image, number of photos, number bids and a see listings button should be attached to this list which takes the selected category as parameter and sends it to the listings page. Search results page should contain a section having a list of related queries according to the queries text and has a button attached to this list which eventually saves the results and collaterals and redirects the page to itself. Search results page should contain a section showing count details having the count value. Search results page should contain a section showing the list of popular related listings with their names, image, prices and formats. A button should be attached to this list which takes the selected current listing as parameter and sends it to the Listings page. Search results page should be accessible from Master page and itself.

o: LISTING

Listing page should contain a section showing main image details with their images and has two buttons attached to it; Mouse over and full screen. Mouse over button leads the page towards a zoom frame in the main frame Listing and Vendor inside same page which shows the zoomed image. Full screen button leads the page towards Images page. Details page should also contain a section showing list of images and has a Mouse over button attached to it which takes the selected image id as parameter and sends it to the main image section. Listing and Vendor in Details page includes two frames; Zoom and Listings&Vendor. Listings&Vendor frame should contain a section showing details of last bid showing its value. Listings&Vendor frame should also contain a section showing details of Listings showing name, format, description, condition, number of bids, number of sold items, number of watches, shipping, location, delivery options, guarantee and payment information. Listings should have tow buttons; add to watch list and add to collection attached to it. Listings&Vendor frame should contain a section showing details of sale price. Listings&Vendor frame should also contain a section showing Vendor details with user name, photo and score. Visit store, more items and follow buttons should also be attached to this section. Details page should be accessible from Search results page.

2) MODELING

MBUITC tool provides the option to open IFML Editor. Using IFML Editor, Eclipse environment is opened and IFML model can be designed by creating a new project for IFML modeling. For designing an IFML model, UML domain model is mandatory which is UML class diagram containing the domain concepts as classes, their attributes and interactions and can easily be created in Eclipse Papyrus editor.

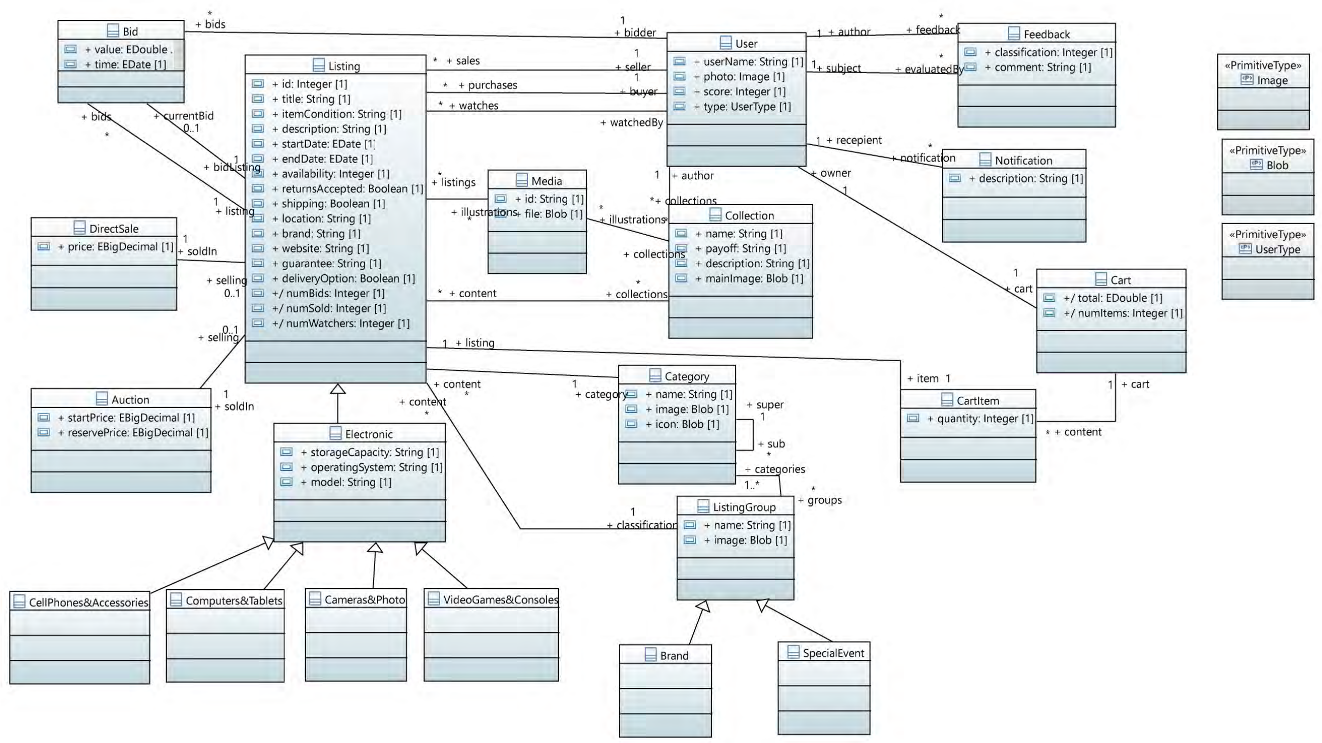


FIGURE 5. Online auctions domain model.

Domain model of the Online Auctions case study is shown in **Fig. 5**.

Domain model of Online Auctions is designed as a UML class diagram in Papyrus editor using Eclipse. Three main classes used in Online Auctions web application are users, bids and listings. Data about user like user name, photo, score and type are kept in User class s attributes. Bids have a value and timestamp. Listings are the main objects. Listings contain attributes like id, title, itemCondition (i.e. new or second hand), description (detailed text), start and end dates (validity period), returnsAccepted, location, shipping (delivery options), currency, guarantee (terms on guarantee offered). Relationships between all the classes have also been shown.

IFML model can be modeled by selecting the domain model as pre-requisite. This helps in the use of domain feature concepts in IFML model. Defined specifications lead us directly to the modeling of IFML model. Master page concept in web is used to avoid duplication. So, it has to be inferred that any page which is master page will be accessible from all other pages. The model developed for online auctions case study is huge and complex. So, to avoid complexity, we represent it as parts of IFML diagram of Online Auctions case study as shown in **Fig. 6**, **Fig. 7**, **Fig. 8**, **Fig. 9** and **Fig. 10**.

3) NAVIGATION MODEL AND TEST CASE GENERATION

In **Fig. 11**, UML and IFML model of online auctions web application are given as input. The transformation does not involve UML model, just IFML model is needed. Eclipse does not allow IFML modeling unless UML domain model is provided for it. UML model with .uml extension is selected and model with .core extension is selected as IFML model. We have provided a folder on desktop as destination. On clicking the Generate button, the input IFML model is transformed into four outputs.

- First output obtained is a simple State Transition Matrix in excel format (.xls). This matrix contains all the view containers of Online Auctions model as states and provides true and false values against their navigation with other states. (A small part of matrix is shown in **Fig. 12**)
- Another matrix generated as output is a detailed Source&Target Information matrix for Online Auctions model in excel format (.xls). Each row of this matrix represents a detailed navigation test case. (A small part of matrix is shown in **Fig. 13**)
- Test case document (.txt) with detailed UI test cases related to Forms, Lists and Details is generated. A total of 41 test cases were generated for Online Auctions IFML model. (One test case from the Test case document is shown in **Fig. 14**)
- A text file (.xta) is generated as last output. This file contains code of navigation model in Timed Automata formalism. This file acts as a template and can be imported in UPPAAL model checker tool for verification of deadlock and reachability. (A portion of xta file is shown in **Fig. 15**)

Status displays that applied transformation was successfully performed.

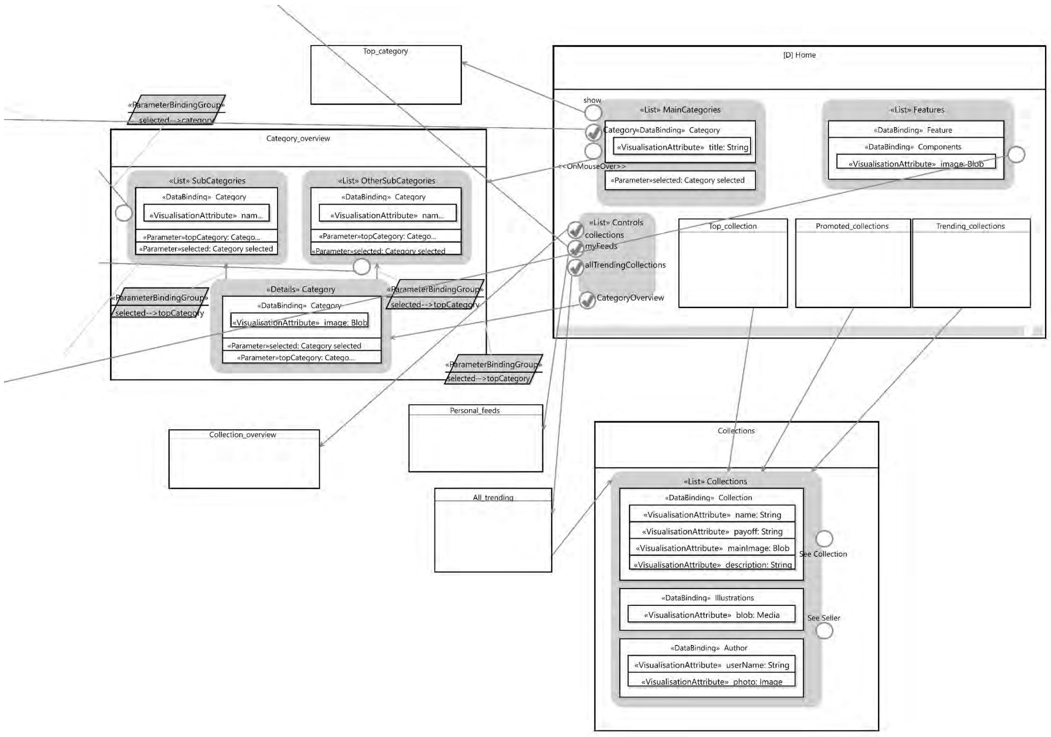

Top_category
[D] Home
show
«List» MainCategories
Category«DataBinding» Category
«VisualisationAttribute» title: String
<<OnMouseOver>>
«Parameter»selected: Category selected
«List» Features
«DataBinding» Feature
«DataBinding» Components
«VisualisationAttribute» image: Blob
«ParameterBindingGroup»
selected-->category
Category_overview
«List» SubCategories
«DataBinding» Category
«VisualisationAttribute» nam...
«Parameter»topCategory: Catego...
«Parameter»selected: Category selected
«List» OtherSubCategories
«DataBinding» Category
«VisualisationAttribute» nam...
«Parameter»topCategory: Catego...
«Parameter»selected: Category selected
«List» Controls
collections
myFeeds
allTrendingCollections
CategoryOverview
Top_collection
Promoted_collections
Trending_collections
«ParameterBindingGroup»
selected-->topCategory
«Details» Category
«DataBinding» Category
«VisualisationAttribute» image: Blob
«Parameter»selected: Category selected
«Parameter»topCategory: Catego...
«ParameterBindingGroup»
selected-->topCategory
«ParameterBindingGroup»
selected-->topCategory
Personal_feeds
Collection_overview
All_trending
Collections
«List» Collections
«DataBinding» Collection
«VisualisationAttribute» name: String
«VisualisationAttribute» payoff: String
«VisualisationAttribute» mainImage: Blob
«VisualisationAttribute» description: String
See Collection
«DataBinding» Illustrations
«VisualisationAttribute» blob: Media
See Seller
«DataBinding» Author
«VisualisationAttribute» userName: String
«VisualisationAttribute» photo: image

FIGURE 6. Online auctions IFML model (diagram 1 of 5).

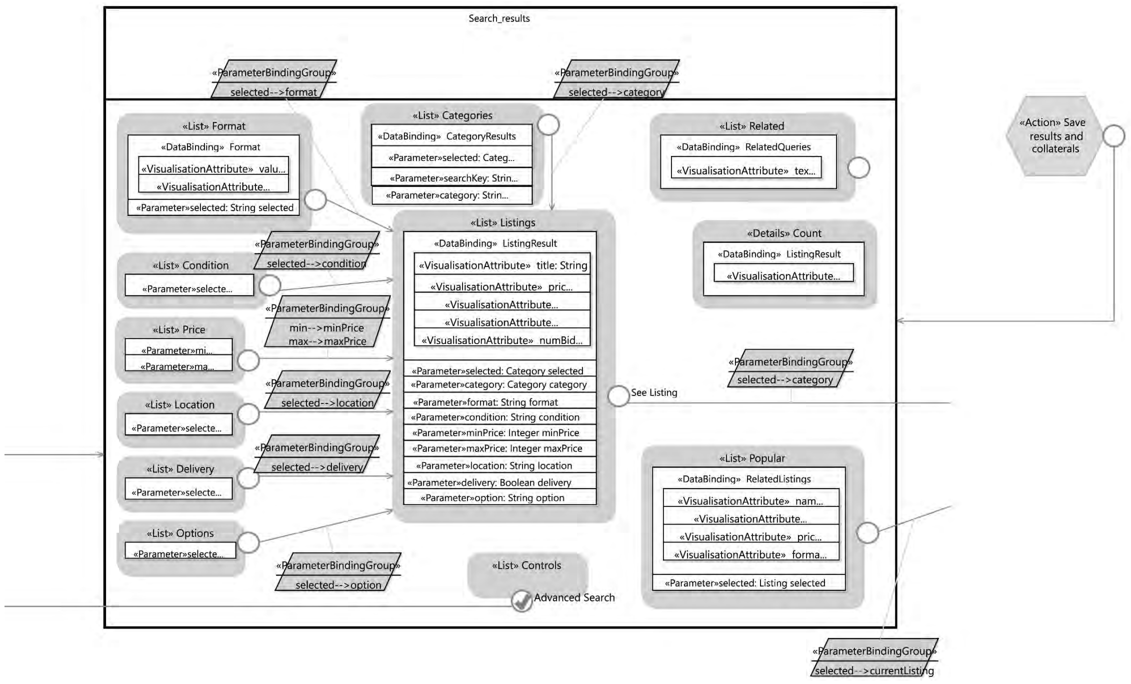

Search_results
«ParameterBindingGroup»
selected-->format
«ParameterBindingGroup»
selected-->category
«List» Format
«DataBinding» Format
«VisualisationAttribute» valu...
«VisualisationAttribute...
«Parameter»selected: String selected
«List» Categories
«DataBinding» CategoryResults
«Parameter»selected: Categ...
«Parameter»searchKey: Strin...
«Parameter»category: Strin...
«List» Related
«DataBinding» RelatedQueries
«VisualisationAttribute» tex...
«Action» Save results and collaterals
«List» Listings
«DataBinding» ListingResult
«VisualisationAttribute» title: String
«VisualisationAttribute» pric...
«VisualisationAttribute...
«VisualisationAttribute...
«VisualisationAttribute» numBid...
«Parameter»selected: Category selected
«Parameter»category: Category category
«Parameter»format: String format
«Parameter»condition: String condition
«Parameter»minPrice: Integer minPrice
«Parameter»maxPrice: Integer maxPrice
«Parameter»location: String location
«Parameter»delivery: Boolean delivery
«Parameter»option: String option
«ParameterBindingGroup»
selected-->condition
«List» Condition
«Parameter»selecte...
«ParameterBindingGroup»
min-->minPrice
max-->maxPrice
«List» Price
«Parameter»mi...
«Parameter»ma...
«ParameterBindingGroup»
selected-->location
«List» Location
«Parameter»selecte...
«ParameterBindingGroup»
selected-->delivery
«List» Delivery
«Parameter»selecte...
«List» Options
«Parameter»selecte...
«ParameterBindingGroup»
selected-->option
«List» Controls
Advanced Search
«Details» Count
«DataBinding» ListingResult
«VisualisationAttribute...
«ParameterBindingGroup»
selected-->category
See Listing
«List» Popular
«DataBinding» RelatedListings
«VisualisationAttribute» nam...
«VisualisationAttribute...
«VisualisationAttribute» pric...
«VisualisationAttribute» forma...
«Parameter»selected: Listing selected
«ParameterBindingGroup»
selected-->currentListing

FIGURE 7. Online auctions IFML model (diagram 2 of 5).

4) AUTOMATED NAVIGATION VERIFICATION

UPPAAL model checker is the tool selected for verification purpose. When we load the xta template file in UPPAAL (**Fig. 16**), it first checks the syntax. Validation in UPPAAL is done using its simulator which makes sure that the selected model is correct and complete. Verification is checked after simulation. Two properties we have considered in verification are:

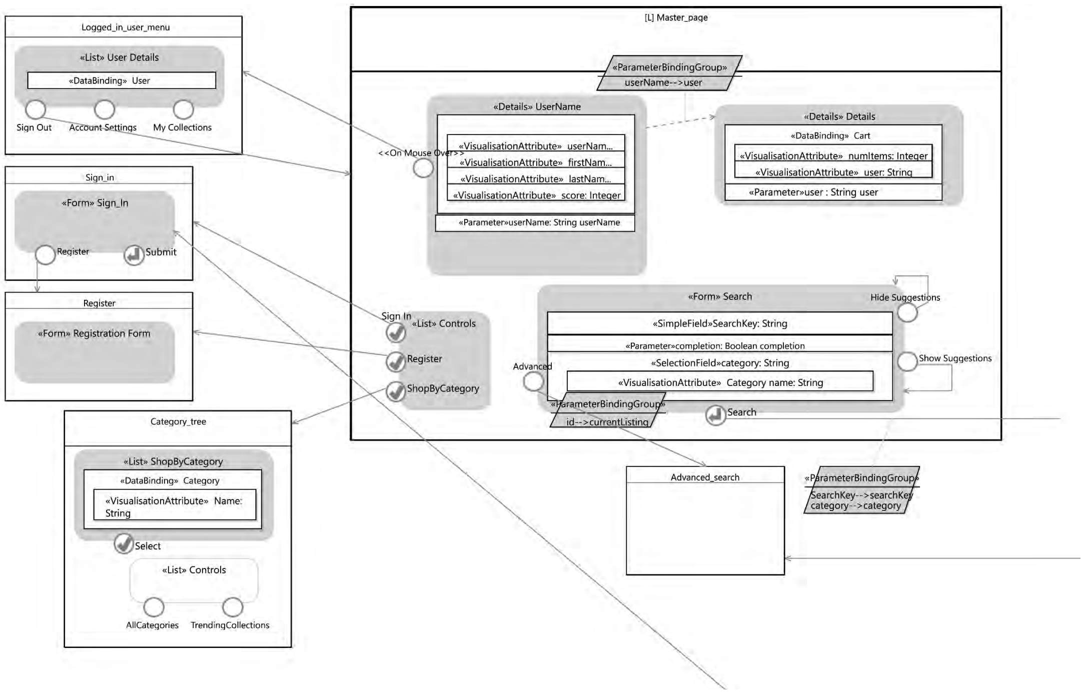

Logged_in_user_menu
«List» User Details
«DataBinding» User
Sign Out
Account Settings
My Collections
Sign_in
«Form» Sign_In
Register
Submit
Register
«Form» Registration Form
Category_tree
«List» ShopByCategory
«DataBinding» Category
«VisualisationAttribute» Name: String
Select
«List» Controls
AllCategories
TrendingCollections
[L] Master_page
«ParameterBindingGroup» userName-->user
«Details» UserName
<<On Mouse Over>>
«VisualisationAttribute» userNam...
«VisualisationAttribute» firstNam...
«VisualisationAttribute» lastNam...
«VisualisationAttribute» score: Integer
«Parameter»userName: String userName
«Details» Details
«DataBinding» Cart
«VisualisationAttribute» numItems: Integer
«VisualisationAttribute» user: String
«Parameter»user : String user
Sign In
«List» Controls
Register
ShopByCategory
«Form» Search
Hide Suggestions
«SimpleField»SearchKey: String
«Parameter»completion: Boolean completion
«SelectionField»category: String
«VisualisationAttribute» Category name: String
Show Suggestions
Advanced
«ParameterBindingGroup» id-->currentListing
Search
Advanced_search
«ParameterBindingGroup» SearchKey-->searchKey category-->category

FIGURE 8. Online auctions IFML model (diagram 3 of 5).

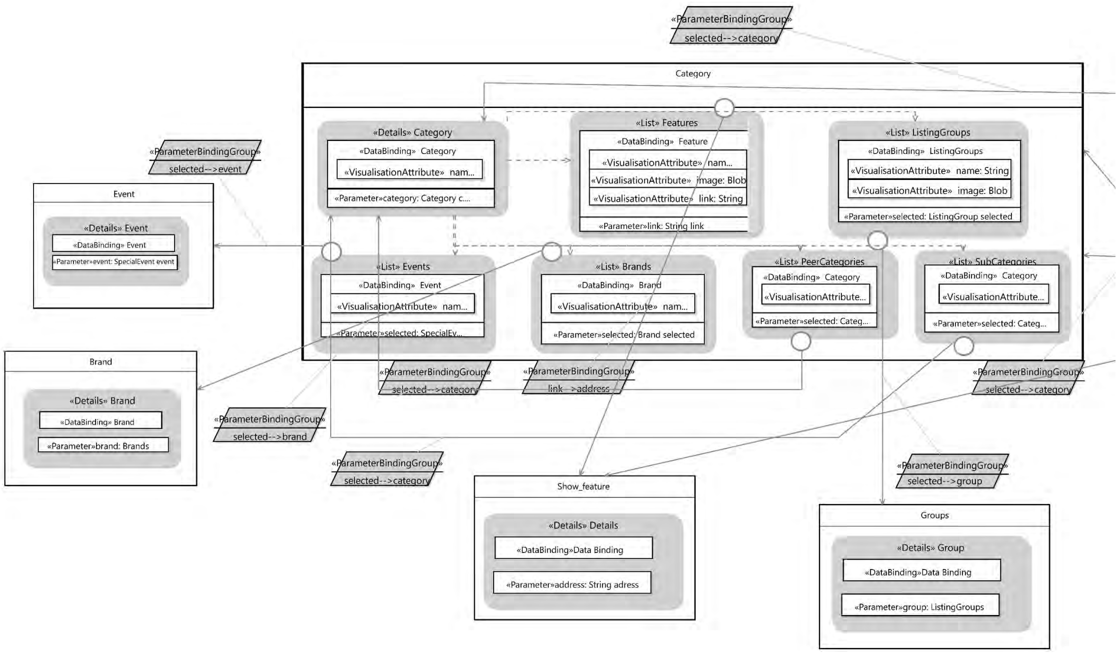

«ParameterBindingGroup» selected-->category
Category
«Details» Category
«DataBinding» Category
«VisualisationAttribute» nam...
«Parameter»category: Category c...
«List» Features
«DataBinding» Feature
«VisualisationAttribute» nam...
«VisualisationAttribute» image: Blob
«VisualisationAttribute» link: String
«Parameter»link: String link
«List» ListingGroups
«DataBinding» ListingGroups
«VisualisationAttribute» name: String
«VisualisationAttribute» image: Blob
«Parameter»selected: ListingGroup selected
«ParameterBindingGroup» selected-->event
Event
«Details» Event
«DataBinding» Event
«Parameter»event: SpecialEvent event
«List» Events
«DataBinding» Event
«VisualisationAttribute» nam...
«Parameter»selected: SpecialEv...
«List» Brands
«DataBinding» Brand
«VisualisationAttribute» nam...
«Parameter»selected: Brand selected
«List» PeerCategories
«DataBinding» Category
«VisualisationAttribute...
«Parameter»selected: Categ...
«List» SubCategories
«DataBinding» Category
«VisualisationAttribute...
«Parameter»selected: Categ...
Brand
«Details» Brand
«DataBinding» Brand
«Parameter»brand: Brands
«ParameterBindingGroup» selected-->category
«ParameterBindingGroup» link-->address
«ParameterBindingGroup» selected-->category
«ParameterBindingGroup» selected-->brand
«ParameterBindingGroup» selected-->category
«ParameterBindingGroup» selected-->group
Show_feature
«Details» Details
«DataBinding»Data Binding
«Parameter»address: String adress
Groups
«Details» Group
«DataBinding»Data Binding
«Parameter»group: ListingGroups

FIGURE 9. Online Auctions IFML model (diagram 4 of 5).

- Reachability: Reachability is checking if the state mentioned in query is reachable from the initial state through any possible sequence (at least one). Reachability makes sure that the web pages are accessible. Query used for checking reachability in UPPAAL model is:

$$E <> Process.Master\_page$$

If the property is satisfied, it means that the ''Master_page'' is reachable from initial page ''Home'' using at least one path.

- Deadlock: Deadlock checking is basically checking if there is a single point in the model which blocks the transition. In simple words, deadlock occurs when there is at least one state that has no next state to go to.

$$A[]\ not\ deadlock$$

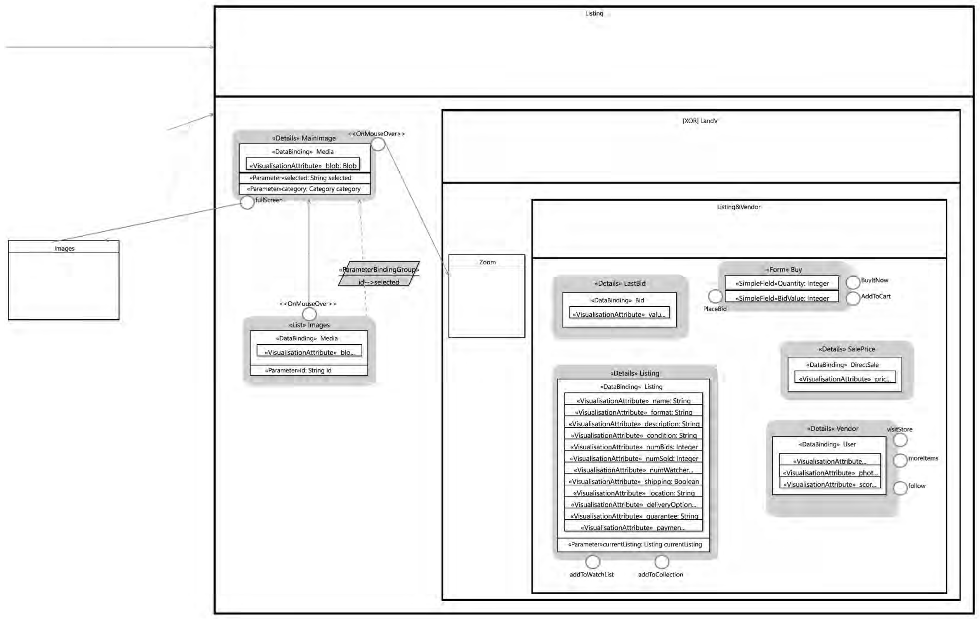
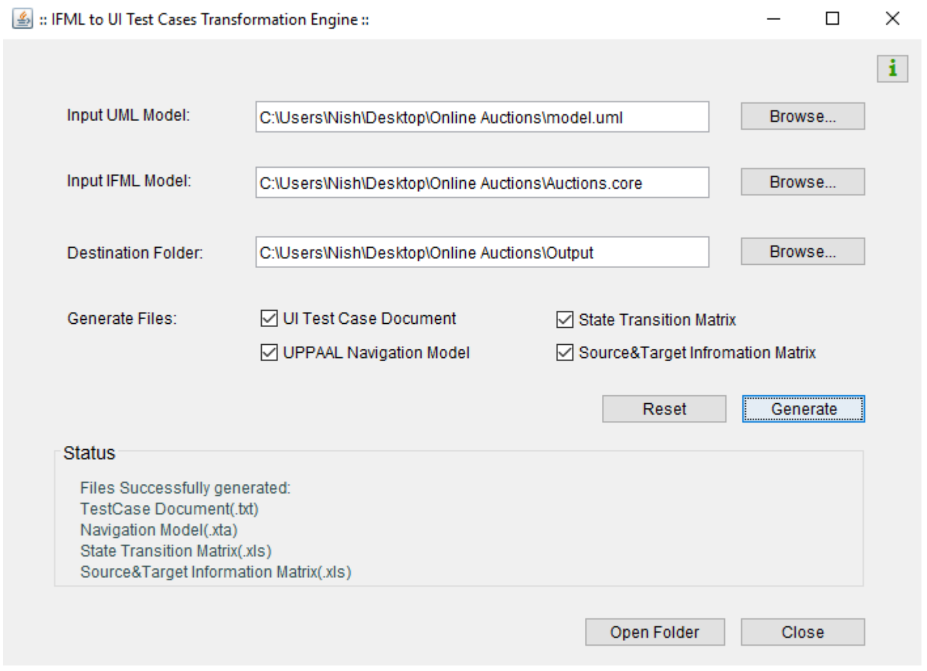

Listing
[XOR] LandV
Listing&Vendor
«Details» MainImage
«DataBinding» Media
«VisualisationAttribute» blob: Blob
«Parameter»selected: String selected
«Parameter»category: Category category
<<OnMouseOver>>
fullScreen
Images
Zoom
«ParameterBindingGroup»
id-->selected
<<OnMouseOver>>
«List» Images
«DataBinding» Media
«VisualisationAttribute» blo...
«Parameter»id: String id
«Details» LastBid
«DataBinding» Bid
«VisualisationAttribute» valu...
«Form» Buy
«SimpleField»Quantity: Integer
«SimpleField»BidValue: Integer
PlaceBid
BuyItNow
AddToCart
«Details» SalePrice
«DataBinding» DirectSale
«VisualisationAttribute» pric...
«Details» Listing
«DataBinding» Listing
«VisualisationAttribute» name: String
«VisualisationAttribute» format: String
«VisualisationAttribute» description: String
«VisualisationAttribute» condition: String
«VisualisationAttribute» numBids: Integer
«VisualisationAttribute» numSold: Integer
«VisualisationAttribute» numWatcher...
«VisualisationAttribute» shipping: Boolean
«VisualisationAttribute» location: String
«VisualisationAttribute» deliveryOption...
«VisualisationAttribute» guarantee: String
«VisualisationAttribute» paymen...
«Parameter»currentListing: Listing currentListing
addToWatchList
addToCollection
«Details» Vendor
«DataBinding» User
«VisualisationAttribute...
«VisualisationAttribute» phot...
«VisualisationAttribute» scor...
visitStore
moreItems
follow

FIGURE 10. Online auctions IFML model (Diagram 5 of 5).



Input UML Model: C:\Users\Nish\Desktop\Online Auctions\model.uml

Input IFML Model: C:\Users\Nish\Desktop\Online Auctions\Auctions.core

Destination Folder: C:\Users\Nish\Desktop\Online Auctions\Output

Generate Files: ☑ UI Test Case Document ☑ State Transition Matrix ☑ UPPAAL Navigation Model ☑ Source&Target Infromation Matrix

Status

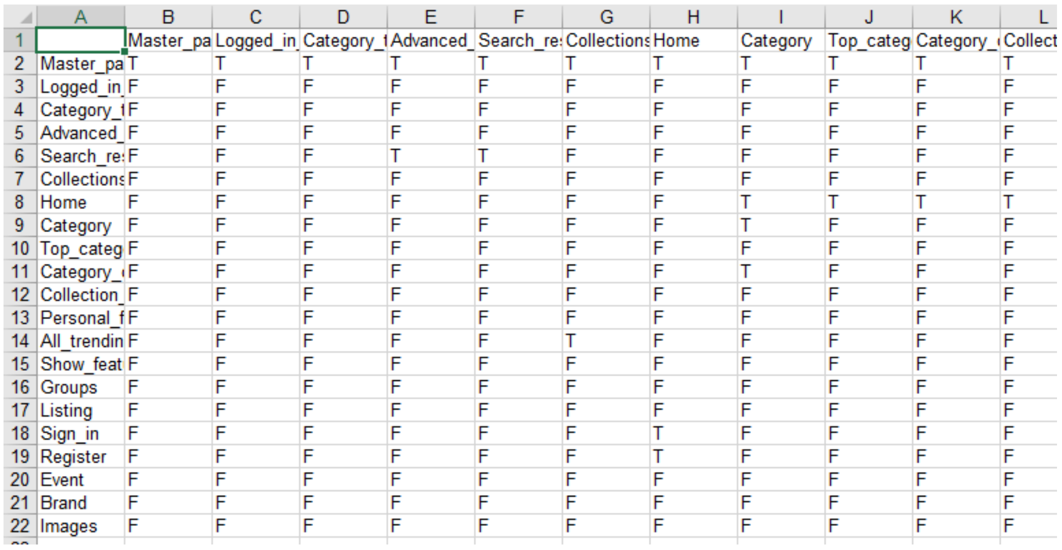
```
Files Successfully generated:
TestCase Document(.txt)
Navigation Model(.xta)
State Transition Matrix(.xls)
Source&Target Information Matrix(.xls)
```

FIGURE 11. Transformation for online auctions model.

If the property is satisfied, it means that the model is deadlock free. **Fig. 17** shows that our output model satisfies both these properties. Hence, our model is reachable and deadlock free.

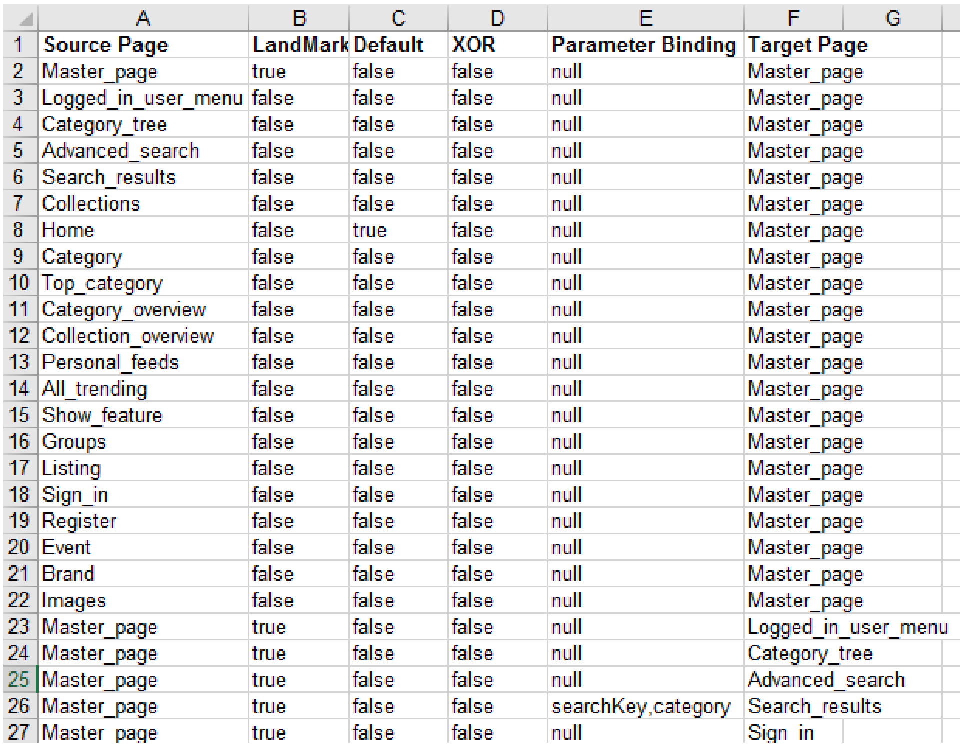
The complete source code of MBUITC Generator tool along with user manual and sample case studies can be found at [32].

Another case regarding the same case study can be discussed where there exist some problems in the design of the web application under discussion. **Fig. 18** elicits the model where some of the pages are unreachable. It can be analyzed that some pages i.e. Images and Collection_overview do not have any outgoing navigational flows eventually should result in a deadlock. **Fig. 19** shows that our output model does not



	A	B	C	D	E	F	G	H	I	J	K	L
1		Master_pa	Logged_in	Category_t	Advanced_	Search_re	Collections	Home	Category	Top_categ	Category_	Collect
2	Master_pa	T	T	T	T	T	T	T	T	T	T	T
3	Logged_in	F	F	F	F	F	F	F	F	F	F	F
4	Category_t	F	F	F	F	F	F	F	F	F	F	F
5	Advanced_	F	F	F	F	F	F	F	F	F	F	F
6	Search_re	F	F	F	T	T	F	F	F	F	F	F
7	Collections	F	F	F	F	F	F	F	F	F	F	F
8	Home	F	F	F	F	F	F	F	T	T	T	T
9	Category	F	F	F	F	F	F	F	T	F	F	F
10	Top_categ	F	F	F	F	F	F	F	F	F	F	F
11	Category_	F	F	F	F	F	F	F	T	F	F	F
12	Collection_	F	F	F	F	F	F	F	F	F	F	F
13	Personal_f	F	F	F	F	F	F	F	F	F	F	F
14	All_trendin	F	F	F	F	F	T	F	F	F	F	F
15	Show_feat	F	F	F	F	F	F	F	F	F	F	F
16	Groups	F	F	F	F	F	F	F	F	F	F	F
17	Listing	F	F	F	F	F	F	F	F	F	F	F
18	Sign_in	F	F	F	F	F	F	T	F	F	F	F
19	Register	F	F	F	F	F	F	T	F	F	F	F
20	Event	F	F	F	F	F	F	F	F	F	F	F
21	Brand	F	F	F	F	F	F	F	F	F	F	F
22	Images	F	F	F	F	F	F	F	F	F	F	F

FIGURE 12. State transition matrix.



	A	B	C	D	E	F	G
1	**Source Page**	**LandMark**	**Default**	**XOR**	**Parameter Binding**	**Target Page**	
2	Master_page	true	false	false	null	Master_page	
3	Logged_in_user_menu	false	false	false	null	Master_page	
4	Category_tree	false	false	false	null	Master_page	
5	Advanced_search	false	false	false	null	Master_page	
6	Search_results	false	false	false	null	Master_page	
7	Collections	false	false	false	null	Master_page	
8	Home	false	true	false	null	Master_page	
9	Category	false	false	false	null	Master_page	
10	Top_category	false	false	false	null	Master_page	
11	Category_overview	false	false	false	null	Master_page	
12	Collection_overview	false	false	false	null	Master_page	
13	Personal_feeds	false	false	false	null	Master_page	
14	All_trending	false	false	false	null	Master_page	
15	Show_feature	false	false	false	null	Master_page	
16	Groups	false	false	false	null	Master_page	
17	Listing	false	false	false	null	Master_page	
18	Sign_in	false	false	false	null	Master_page	
19	Register	false	false	false	null	Master_page	
20	Event	false	false	false	null	Master_page	
21	Brand	false	false	false	null	Master_page	
22	Images	false	false	false	null	Master_page	
23	Master_page	true	false	false	null	Logged_in_user_menu	
24	Master_page	true	false	false	null	Category_tree	
25	Master_page	true	false	false	null	Advanced_search	
26	Master_page	true	false	false	searchKey,category	Search_results	
27	Master_page	true	false	false	null	Sign_in	

FIGURE 13. Source& target matrix.

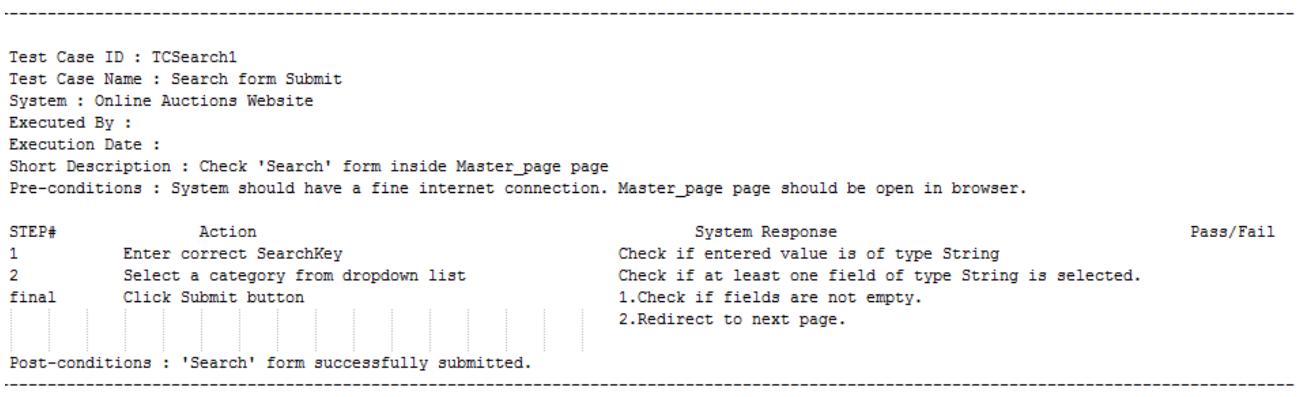
satisfy the reachability and deadlock property for this case, which demonstrates the validity of the approach.

B. LIBRARY CASE STUDY

We present a chunk of Library case study as second case study for validation to show that our proposed approach works for case studies of all sizes. **Section V-B-1** covers the requirement specification for a part of library web application. **Section V-B-2** and **Section V-B-3** present the UML domain modeling and IFML modeling of this case study in Eclipse editor using the IFML plugin, respectively. Lastly, the transformation and verification of the case study modeled in IFML has been provided in **Section V-B-4.**

1) REQUIREMENT SPECIFICATION

Following are the details of the main web pages and their specifications included in the library case study. Five important pages have been selected in the case study which specify a simple behavior of adding a book.



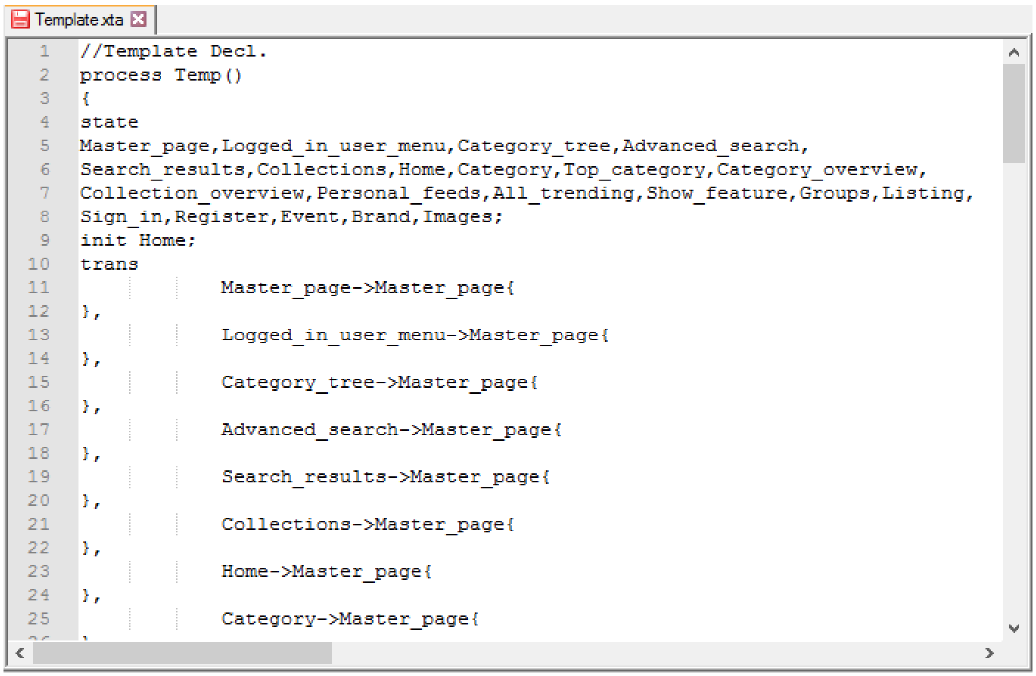
```
------------------------------------------------------------------------------------------------------------------------

Test Case ID : TCSearch1
Test Case Name : Search form Submit
System : Online Auctions Website
Executed By :
Execution Date :
Short Description : Check 'Search' form inside Master_page page
Pre-conditions : System should have a fine internet connection. Master_page page should be open in browser.

STEP#           Action                                          System Response                              Pass/Fail
1           Enter correct SearchKey                         Check if entered value is of type String
2           Select a category from dropdown list            Check if at least one field of type String is selected.
final       Click Submit button                             1.Check if fields are not empty.
                                                            2.Redirect to next page.

Post-conditions : 'Search' form successfully submitted.
------------------------------------------------------------------------------------------------------------------------
```

FIGURE 14. Test case for search form.



Template.xta

```
//Template Decl.
process Temp()
{
state
Master_page,Logged_in_user_menu,Category_tree,Advanced_search,
Search_results,Collections,Home,Category,Top_category,Category_overview,
Collection_overview,Personal_feeds,All_trending,Show_feature,Groups,Listing,
Sign_in,Register,Event,Brand,Images;
init Home;
trans
            Master_page->Master_page{
},
            Logged_in_user_menu->Master_page{
},
            Category_tree->Master_page{
},
            Advanced_search->Master_page{
},
            Search_results->Master_page{
},
            Collections->Master_page{
},
            Home->Master_page{
},
            Category->Master_page{
```

FIGURE 15. Navigation model xta file.

a: HOME

Home page contains a section in which recently published books are shown in a list along-with their titles and publication years. On selecting any book, it leads us to Book Details page.

b: BOOK DETAILS

Book Details page has a section which shows details of the selected book. The details about title, author name, publication year and description about the book are shown.

c: BOOK LIST

Book List shows a list of books along-with their title, author name, year and description. On selecting a book, the page is navigated to Book Details page. Another button named "Add book" should also be attached which leads towards Add Book page and selected book is passed as parameter.

d: ADD BOOK

This page contains a simple form for adding a book. The form takes title, author year and description as input and on clicking Submit button, it saves the data to the Book List.

e: ERROR PAGE

If any error occurs during saving the data in Book List, then Error page is displayed.

2) MODELING

As we have selected a simple feature of the case study, only one class is needed in domain model. Book class contains

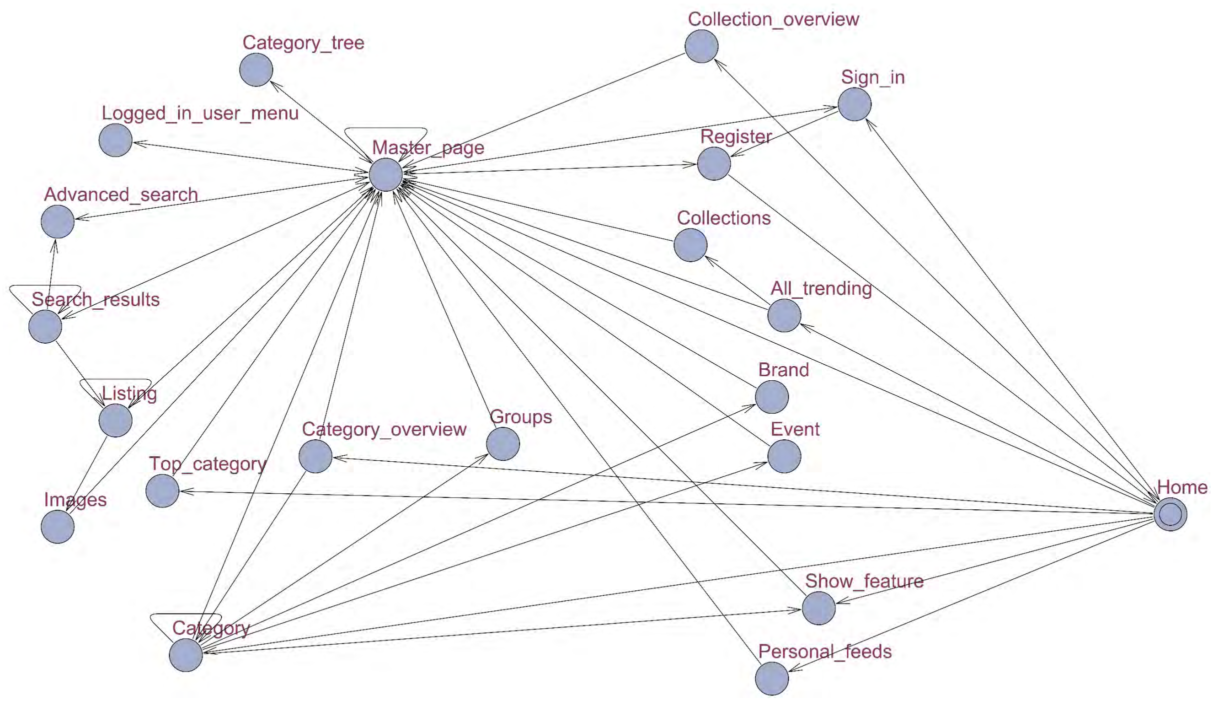
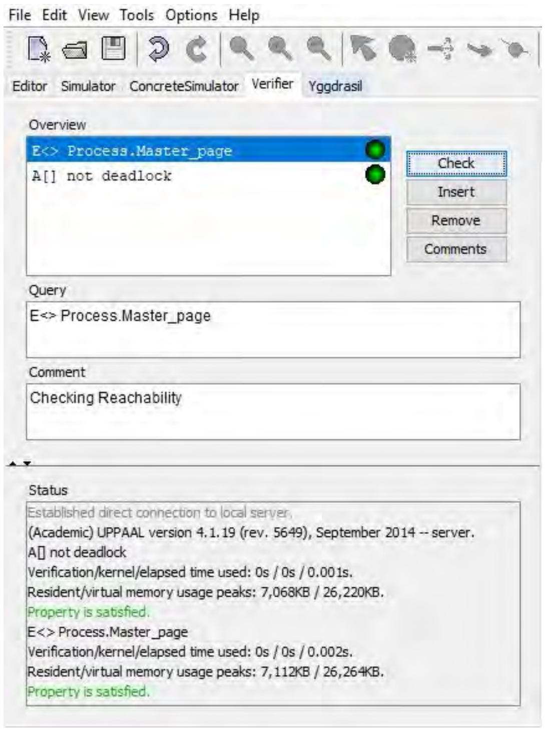


FIGURE 16. UPPAAL model for online auctions.


File Edit View Tools Options Help
Editor Simulator ConcreteSimulator Verifier Yggdrasil
Overview
Check
Insert
Remove
Comments
Query
Comment
Status

```
E<> Process.Master_page
A[] not deadlock
```

Query: E<> Process.Master_page

Comment: Checking Reachability

```
Established direct connection to local server.
(Academic) UPPAAL version 4.1.19 (rev. 5649), September 2014 -- server.
A[] not deadlock
Verification/kernel/elapsed time used: 0s / 0s / 0.001s.
Resident/virtual memory usage peaks: 7,068KB / 26,220KB.
Property is satisfied.
E<> Process.Master_page
Verification/kernel/elapsed time used: 0s / 0s / 0.002s.
Resident/virtual memory usage peaks: 7,112KB / 26,264KB.
Property is satisfied.
```

FIGURE 17. Deadlock and reachability verification.

important data about book i.e. title, author, year and description. These attributes are later used in the IFML model. Book class shown in **Fig. 20** has been designed using Papyrus editor.

To model the IFML diagram for library system, its UML model is taken as pre-requisite and IFML model is designed in Eclipse IFML editor taking in account the specifications given. The pages mentioned in specifications are modeled as view containers. Lists, details and forms are modeled as extensions of view components.**Fig. 21** shows the IFML model for the part selected from library case study.

3) NAVIGATION MODEL AND TEST CASE GENERATION

Library UML (.uml) and IFML (.core) models are given as input and a path for destination folder is provides. When we click on ‘‘Generate’’ button, the input IFML model is transformed into State Transition Matrix, Source&Target Information Matrix, a test case document (one test case is shown in **Fig. 22**) and an xta file (**Fig. 23**) containing code in Timed Automata formalism that can be opened in UPPAAL.

4) AUTOMATED NAVIGATION VERIFICATION

When the xta template file is loaded in UPPAAL, it shows us the navigation model for library system (**Fig. 24**). Reachability and deadlock properties are verified for library system using UPPAAL property checking syntax.

The complete source code of MBUITC Generator tool along with user manual and sample case studies can be found at [32].

VI. COMPARATIVE ANALYSIS WITH STATE-OF-THE-ART

In this section, we perform a comparative analysis with the state-of-the-art in order to highlight the significance of the work done. On whole, we have identified six research works

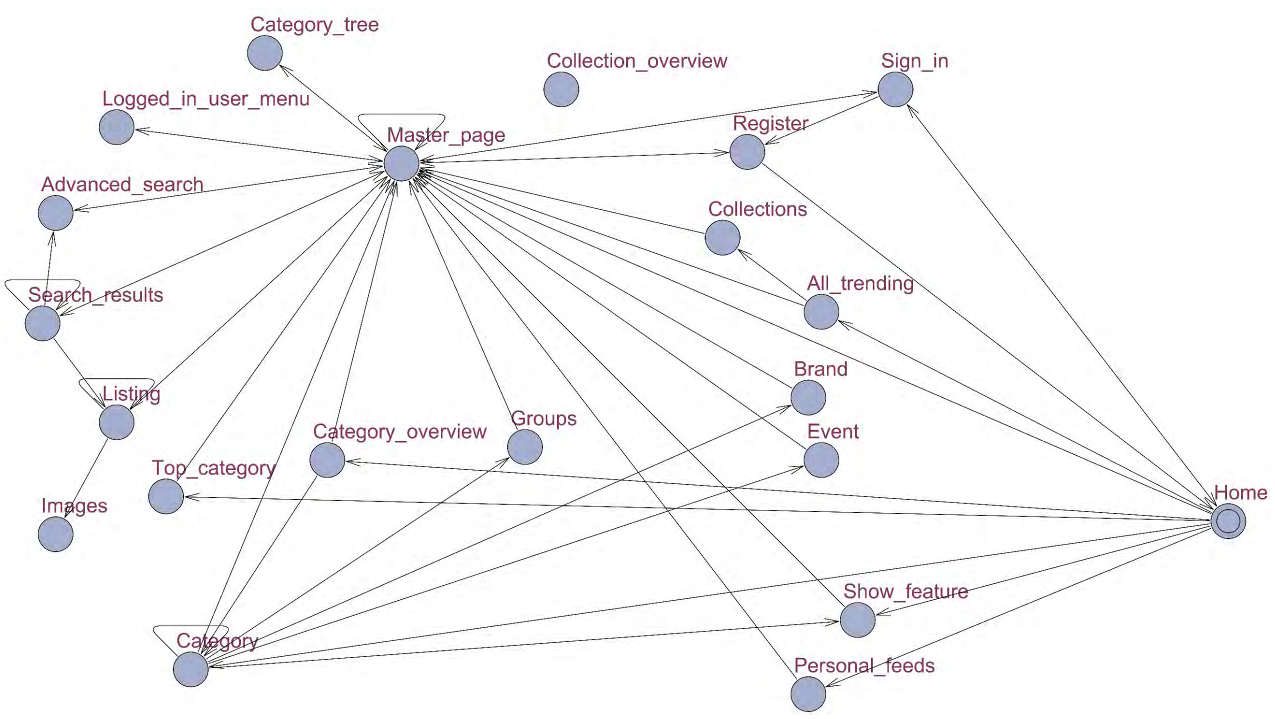
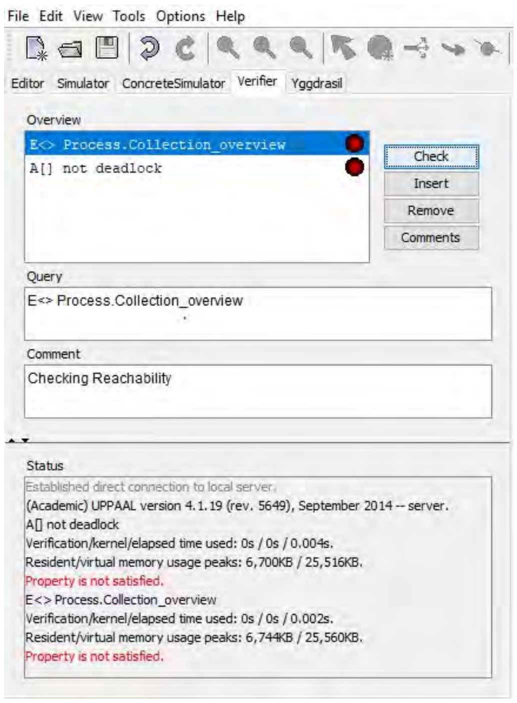


FIGURE 18. UPPAAL model for online auctions case2.


File Edit View Tools Options Help
Editor Simulator ConcreteSimulator Verifier Yggdrasil
Overview
Check
Insert
Remove
Comments
Query
Comment
Status

```
E<> Process.Collection_overview
A[] not deadlock
```

Query: E<> Process.Collection_overview

Comment: Checking Reachability

```
Established direct connection to local server.
(Academic) UPPAAL version 4.1.19 (rev. 5649), September 2014 -- server.
A[] not deadlock
Verification/kernel/elapsed time used: 0s / 0s / 0.004s.
Resident/virtual memory usage peaks: 6,700KB / 25,516KB.
Property is not satisfied.
E<> Process.Collection_overview
Verification/kernel/elapsed time used: 0s / 0s / 0.002s.
Resident/virtual memory usage peaks: 6,744KB / 25,560KB.
Property is not satisfied.
```

FIGURE 19. Deadlock and reachability verification: case2.

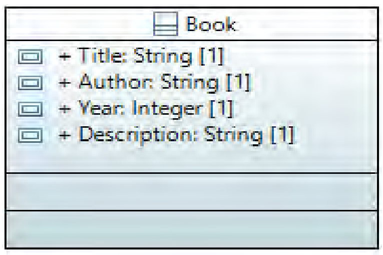


FIGURE 20. Library domain model.

related to model-based UI test case generation. **Table 5** presents an overall comparative analysis of our proposed approach with the state of art. We have selected six parameters for comparison; ***1) Reference #*** is used to represent he reference number of the selected work, ***2) Test Execution*** is used to represent whether the selected research provides execution of the derived testing artifacts. This parameter can have three values; Yes, No or Not Applicable (in case no information is given), ***3) Modeling Language*** is used to indicate the language used for modeling, ***4) Tool Support*** available for the proposed approach is represented as either complete, partial or not available, ***5) Testing Aspects*** indicate that either functional or navigation testing has been focused.

In **Table 5**, six research works have been selected as references excluding the extended work of the research studies. From the literature review we have identified six researches which have worked exclusively on model-based UI test case generation. From which three research works were based on UML modeling, two on IFML and one on BPM (Business Process Modeling). We have observed that no research related to model-based UI generation from WebML has been identified. Main reason to which can be that WebML was used for a short duration and it was not a standard language and soon got transformed into IFML as a standard modeling

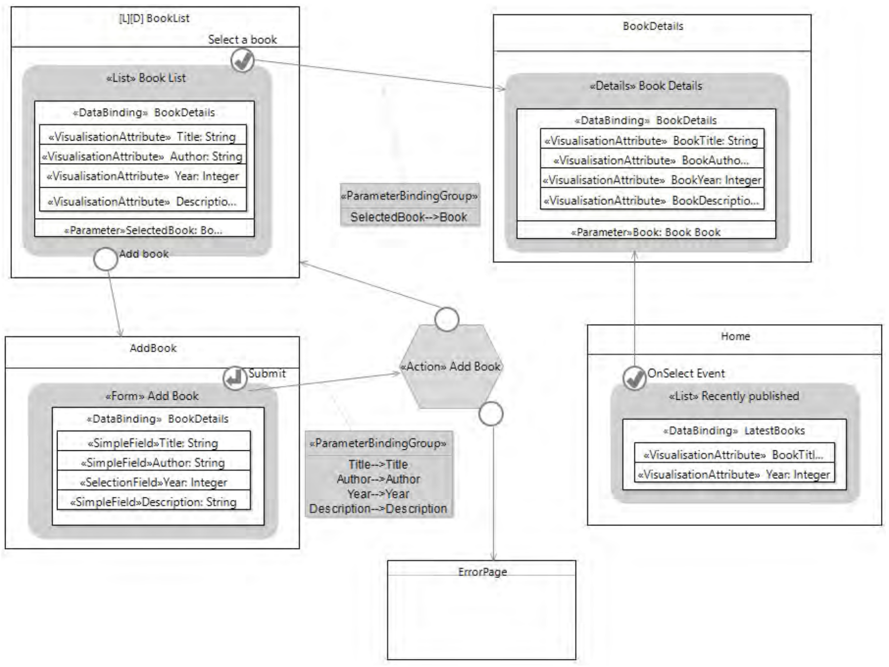
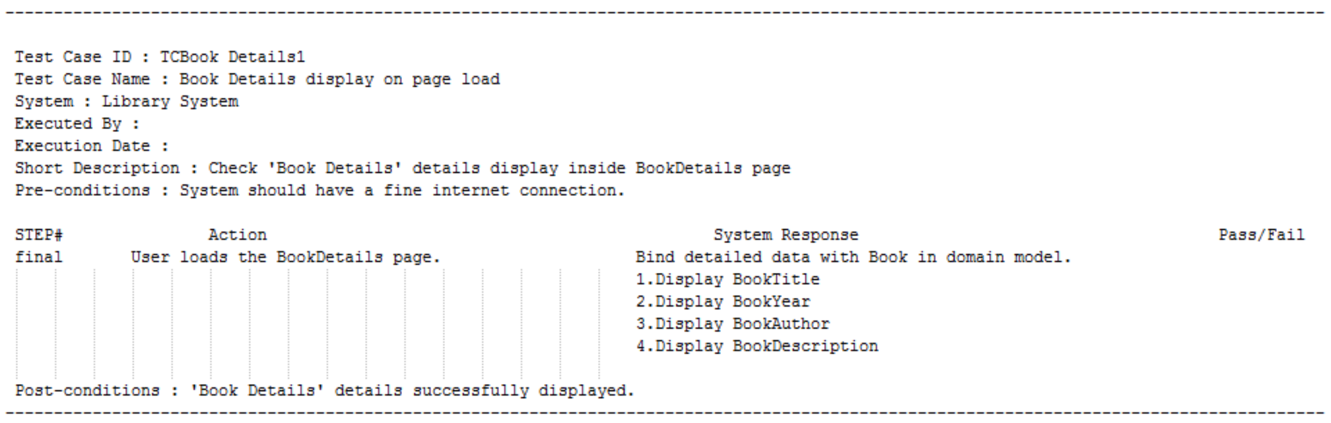

[L][D] BookList
Select a book
«List» Book List
«DataBinding» BookDetails
«VisualisationAttribute» Title: String
«VisualisationAttribute» Author: String
«VisualisationAttribute» Year: Integer
«VisualisationAttribute» Descriptio...
«Parameter»SelectedBook: Bo...
Add book
«ParameterBindingGroup»
SelectedBook-->Book
BookDetails
«Details» Book Details
«DataBinding» BookDetails
«VisualisationAttribute» BookTitle: String
«VisualisationAttribute» BookAutho...
«VisualisationAttribute» BookYear: Integer
«VisualisationAttribute» BookDescriptio...
«Parameter»Book: Book Book
AddBook
Submit
«Form» Add Book
«DataBinding» BookDetails
«SimpleField»Title: String
«SimpleField»Author: String
«SelectionField»Year: Integer
«SimpleField»Description: String
«Action» Add Book
«ParameterBindingGroup»
Title-->Title
Author-->Author
Year-->Year
Description-->Description
Home
OnSelect Event
«List» Recently published
«DataBinding» LatestBooks
«VisualisationAttribute» BookTitl...
«VisualisationAttribute» Year: Integer
ErrorPage

FIGURE 21. Library IFML model.


```
----------------------------------------------------------------------------------------------------------------------------------------

 Test Case ID : TCBook Details1
 Test Case Name : Book Details display on page load
 System : Library System
 Executed By :
 Execution Date :
 Short Description : Check 'Book Details' details display inside BookDetails page
 Pre-conditions : System should have a fine internet connection.

 STEP#             Action                                        System Response                               Pass/Fail
 final       User loads the BookDetails page.              Bind detailed data with Book in domain model.
                                                           1.Display BookTitle
                                                           2.Display BookYear
                                                           3.Display BookAuthor
                                                           4.Display BookDescription

 Post-conditions : 'Book Details' details successfully displayed.
----------------------------------------------------------------------------------------------------------------------------------------
```

FIGURE 22. Test case for book details.

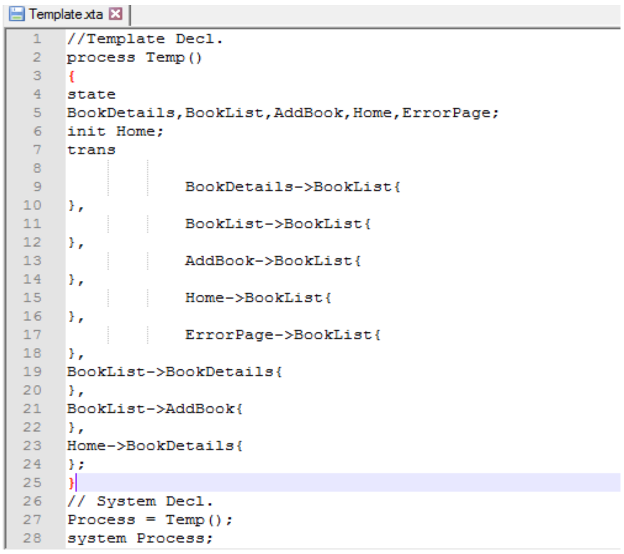
TABLE 5. Overall comparative analysis.

Reference #	Test Execution	Modeling Language	Tool Support	Testing Aspects
[1]	NA	UML	None	Navigation
[2]	Yes	UML	None	Navigation
[7][8]	Yes	IFML	Partial	Functional
[21][22][23]	Yes	IFML	Complete	Navigation
[26]	NA	UML	Partial	Functional
[27]	Yes	BPMN	Partial	Functional
Our Approach	Yes	IFML	Complete	UI Navigation

language for representation of GUI. For example, Suhag and Bhatia et al. [1] worked on MDE along-with data mining techniques on UML web and sequence diagrams in order to obtain a navigation matrix containing the resulting test cases. Matrix parameters i.e. Source page, target page and arguments etc. are obtained using data mining SQL methods. These abstract test cases produced in form of matrix are not executable. Frajtak *et al.* [7], [8] worked on generation

TABLE 6. Comparative analysis with model-based UI test case generation.

Reference #	Automated Test Cases	Navigation Testing	Automation level	Formalism	Applicability
[7][8]	Yes	No	Semi-automated	None	Narrow
[21][22][23]	No	Yes	Fully-automated	Petri nets	Narrow
Our Approach	Yes	Yes	Fully-automated	Timed Automata	Broad



```
//Template Decl.
process Temp()
{
state
BookDetails,BookList,AddBook,Home,ErrorPage;
init Home;
trans

            BookDetails->BookList{
},
            BookList->BookList{
},
            AddBook->BookList{
},
            Home->BookList{
},
            ErrorPage->BookList{
},
BookList->BookDetails{
},
BookList->AddBook{
},
Home->BookDetails{
};
}
// System Decl.
Process = Temp();
system Process;
```

FIGURE 23. Navigation model xta file.

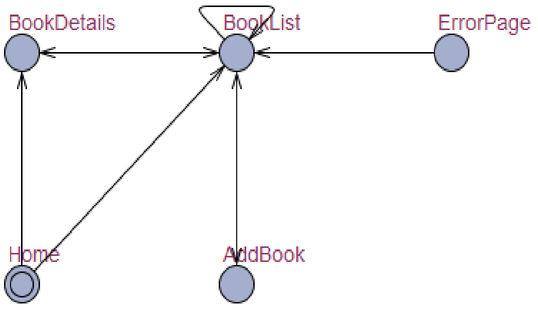


FIGURE 24. UPPAAL model for library system.

of executable test cases in JavaScript format using MDE transformations and a template engine but some of the core IFML concepts are not considered. The tests are executed using Jasmine test runner but no verification has been provided in the research paper. Saleh and Salem [2] used MDE for transition based testing of application GUI models. CTT and ESDM (navigation model) have been used. Model-based transformation have been applied in order to generate the test oracle as transition matrix containing events and states. Only navigation testing has been covered in this research and simulation using EDSM model is also provided. An online open source tool "IFMLEdit.org" has been implemented by Bernaschina *et al.* [23] which provides modeling facility in IFML, generates code in json format using MDE transformations on IFML model, also provides verification for navigation testing using its simulator. Functional testing is not covered in IFMLedit.org. Bowen *et al.* and Reeves [26] worked on interactive systems and focused on test case generation by tight integration of UI and functionality but mainly the assertions are based on functional testing. UI Model was based on finite state automaton, which can be developed using state diagrams in UML. After transformation of abstract tests to concrete tests, executable test oracles are produced. Gupta and Surve *et al.* [27] also worked on both UI and functional test case generation but used BPMN to represent the business flows along with UI but main focus of this work is on fully-automated functional testing.

Table 6 presents the comparative analysis of our proposed work with the researches which worked on model-based UI test case generation using IFML. We have selected five parameters for comparison; ***1) Reference #*** is used to represent he reference number of the selected work, ***2) Automated Test Cases*** represents that if the research resulted in generation of automated test cases or test case documentation, ***3) Navigation Testing*** represents if the test case generation approach has performed navigation testing or has provided state transition or navigation matrix, 4***) Automation Level*** indicates if the navigation testing has been semi or fully automated, ***5) Formalism*** represents the formal verification technique used, ***6) Applicability*** represents the applicability of research in the domain of web application testing. Narrow and Broad indicate the presence and absence of the parameter, respectively.

Table 6 provides a summary of overall studies found on Model-based GUI testing along-with the proposed approach. Frajtak et al. [7], [8] used IFML as modeling language for UI components representation. MDE transformation has been applied in order to generate abstract test case scenarios which are again transformed into specific test case scenarios using the WebdriverIO template. Jasmine test runner then executes this JavaScript code. Even if executable tests are generated on the basis of events, but navigation testing has not been covered. Hence, mentioned as Semi-automated in **Table 6**. Bernaschina *et al.* [23] majorly contributed in code prototype generation for mobile and web applications. It presents an open source online MDD tool called IFMLEdit.org which generated fast prototypes for mobile and web apps using transformations on IFML models. Focus of this research is code generation so testing has not been covered. Although, mapping from IFML to Place Chart Nets (PCN) formalism of Petri Nets, has been applied for model checking which eventually checks navigation in the model i.e. fully-automated navigation verification. Although Bernaschina *et al.* [23]

presented a very good work which illustrated the strength of IFML based modeling for web application code generation, there are few issues in IFML.Edit.org editor. This tool only works on IFML models modeled in its own environment, models generated by using other tools i.e. Eclipse and WebRatio etc. are not supported. Whereas, our proposed approach does not depend on any tool. IFML core model modeled in any tool can be loaded and tested. It provides an option to model IFML models in Eclipse environment and also welcomes the model already generated in any other environment with .core extension. IFML.Edit.org does not support domain modeling and data type. Metamodel concepts are not incorporated fully in their approach which caused mismatch of concepts.

Our proposed approach uses IFML based modeling and provides the complete UI testing including the automated test cases and the navigation verification after successful transformations on IFML model. Meanwhile, it also provides simulation, reachability and deadlock freedom verification for the navigation model using Timed Automata in UPPAAL tool.

VII. DISCUSSION AND LIMITATIONS

From the literature review, it can be analyzed that there is a very limited amount of research work done on IFML in the area of model-based UI testing. In addition, the available research work does not capture complete UI related testing. Most of the work has been done only for navigation testing defocusing the testing needed for UI elements structure and the content to be displayed. Our proposed approach is a first step toward automated UI testing using IFML including the navigation testing as well as the content testing. The approach generates test cases using model-based testing technique on UI modeling of the web application using IFML. The tool we have built, produces all broad testing artifacts that can be used in the testing process of different industries. Motivation behind this work is to provide early test cases so that quality can be build inside the application which eventually proved out to be a cost and time efficient approach. The generated test cases provide complete and detailed UI testing for example, how will the information flow take place and what information is passed during navigation? Such details are available in the Source and Target matrix produced by the MBUITC tool. The state transition matrix focuses entirely on the control flow and linking of pages. Furthermore, the test cases related to the complete structure displayed in web pages of the web application are provided in the test case document. Meanwhile, navigation model is provided in language which is supported by UPPAAL which can simulate the model and verify its reachability and deadlock freedom. Test cases generated are abstract and not directly executable.

It is also important to mention here that IFML has been inspired from WebML but is not fully evolved yet. Therefore, the tool support for IFML is limited at the moment. For example, only two reliable tools (i.e. WebRatio and Eclipse IFML plugin) are available that provide support for IFML modeling. However, both tools are not mature enough for IFML modeling and there are some problems encountered during model development. For example, the major issue is the support for IFML nested containers as both tools do not allow nesting for more than 3 levels at the moment. Even though the events are allowed on view containers, but both tools do not allow them as well. Similarly, navigation flows are not allowed inside nesting hierarchy. There are few other limitations to IFML modeling found using these tools but as the IFML is evolving and a lot of work is being done on its tool support, we are hoping that these issues will be resolved soon. That's why MBUITC tool is developed with generic approach that will be worked well after the resolution of aforementioned issues in existing IFML tools.

It can be argued that abstract test cases generated in result of the proposed approach will require some additional work to execute but, at the platform independent level, more detailed and specific test cases cannot be generated, it would require some part of implementation. However, the abstract test cases generated will help the developers and testers. For developers, the testing documents will help to clarify what arguments or information flow needs to be passed during navigation. For testers, this document can be helpful in a way that there would be no effort required in process of creating separate test cases for each web page element and following a consistent format. The abstract test cases can help for manual testing as well as in using the automated testing tools i.e. Selenium etc.

Two case studies of different sizes have been selected in order to validate our proposed approach. First case study is a very detailed description of Online Auctions web application which included more than 25 view containers and plentiful view components and navigation flows. Whereas, the Library case study was a small but main part of Library system which included 5 view containers, 4 view components and various navigation flows. A total of 41 test cases were generated for whole Online Auctions IFML model whereas, only 4 test cases were generated for Library IFML model as it was only a small part of whole model. The purpose of taking into account these two case studies is to validate the MBUITC tool for both large and small scale web applications. As we have taken the first step for automated testing of IFML models, there are a few limitations to our work. For example, we have currently only selected limited core metamodel elements e.g. *ViewContaier*, *ViewComponent*, *NavigationFlow* etc. and a few important extensions metamodel elements e.g. *Form*, *List* and *Details* etc. Although selected IFML constructs are sufficient for modeling large scale web applications, there are many other IFML constructs that we have not yet considered in this work e.g. *Menu*, *Window*, *Module* and *ActivationExpression* etc. The proposed approach and the developed tool (MBUITC) are highly scalable. For example, it is fairly possible to extend the proposed approach for behavioral testing too. To achieve this, UML behavior diagrams can be integrated with IFML models to generate fully functional test cases for business processes along-with the existing WUI testing artifacts. In addition, more IFML constructs (e.g. *Menu*,Action, *Window*, *Module, Context* and *ActivationExpression* etc.) can be considered in the proposed approach. On the other hand, the functions of open source MBUITC Generator tool can be enhanced with ease. For example, more transformation procedures can be implemented by only amending main.mtl file. We intend to work on remaining IFML constructs in our next article.

VIII. CONCLUSION AND FUTURE WORK

This article presents a novel approach to automatically generate Web User Interface (WUI) test cases from Interaction Flow Modeling Language (IFML) models. Particularly, a novel model-based methodology is proposed to automatically generate test case document, state transition matrix, source and target information matrix and timed automata navigation models from the initial collected web application requirements which are represented through domain (UML) and front end (IFML) models. As a part of research, Model-based User Interface Test Case (MBUITC) generator tool is implemented in JAVA using Model to Text (M2T) transformation approach through Acceleo. The applicability of proposed approach is validated through two bench mark case studies. The results shown that the proposed approach provides test cases at the early stages of development i.e. specification and analysis, which eventually helps in building a right product at the right time at comparatively lower cost.

Future work includes improving and extending proposed approach in order to support other important IFML constructs like *Module, Action, Menu, Context* and *Expression* etc. Furthermore, the integration of UML behavior diagrams with IFML models will be performed to generate fully functional test cases for business processes along-with the existing WUI testing artifacts.

REFERENCES

[1] V. Suhag and R. Bhatia, "Model based test cases generation for Web applications," *Int. J. Comput. Appl.*, vol. 92, no. 3, pp. 23–31, Apr. 2014.

[2] E. M. Saleh and O. A. S. Salem, "A model-driven engineering transition-based GUI testing technique," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Las Vegas, NV, USA, Dec. 2015, pp. 108–113.

[3] M. W. Anwar, M. Rashid, F. Azam, and M. Kashif, "Model-based design verification for embedded systems through SVOCL: An OCL extension for system verilog," *Des. Automat. Embedded Syst.*, vol. 21, no. 1, pp. 1–36, Mar. 2017.

[4] S. Jácome and J. D. Lara, "Controlling meta-model extensibility in model-driven engineering," *IEEE Access*, vol. 6, pp. 19923–19939, 2018.

[5] S. Roubi, M. Erramdani, and S. Mbarki, "Extending graphical part of the interaction flow modeling language to generate rich Internet graphical user interfaces," in *Proc. 4th Int. Conf. Model-Driven Eng. Softw. Develop. (MODELSWARD)*, Rome, Italy, Feb. 2016, pp. 161–167.

[6] N. Laaz and S. Mbarki, "Integrating IFML models and owl ontologies to derive UIs Web-Apps," in *Proc. Int. Conf. Inf. Technol. Org. Develop. (ITOD)*, Fez, Morocco, Mar./Apr. 2016, pp. 1–6.

[7] K. Frajták, M. Bureš, and I. Jelínek, "Transformation of IFML schemas to automated tests," in *Proc. Conf. Res. Adapt. Convergent Syst.*, Prague, Czech, Oct. 2015, pp. 509–511.

[8] K. Frajták, M. Bureš, and I. Jelínek, "Using the interaction flow modelling language for generation of automated front-end tests," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (ACSIS)*, vol. 6, 2015, pp. 117–122.

[9] M. Brambilla, A. Mauri, M. Franzago, and H. Muccini, "A model-based method for seamless web and mobile experience," in *Proc. 1st Int. Workshop Mobile Develop.*, Amsterdam, Netherlands, Oct. 2016, pp. 33–40.

[10] C. Bernaschina, M. Brambilla, T. Koka, A. Mauri, and E. Umuhoza, "Integrating modeling languages and Web logs for enhanced user behavior analytics," in *Proc. 26th Int. Conf. World Wide Web Companion*, pp. 171–175, Apr. 2017.

[11] R. Acerbis, A. Bongio, S. Butti, and M. Brambilla, "Model-driven development of cross-platform mobile applications with webratio and IFML," in *Proc. 2nd ACM Int. Conf. Mobile Softw. Eng. Syst.*, Florence, Italy, May 2015, pp. 170–171.

[12] R. Acerbis, A. Bongio, M. Brambilla, and S. Butti, "Model-driven development based on omg's IFML with webratio web and mobile platform," in *Proc. Int. Conf. Web Eng.*, Jun. 2015, pp. 605–608.

[13] S. Roubi, M. Erramdani, and S. Mbarki, "A model driven approach to generate graphical user interfaces for rich Internet applications using interaction flow modeling language," in *Proc. ISDA*, Marrakech, Morocco, Dec. 2015, pp. 272–276.

[14] M. Brambilla, A. Mauri, and E. Umuhoza, "Extending the interaction flow modeling language (IFML) for model driven development of mobile applications front end," in *Mobile Web Information Systems* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2014, pp. 176–191.

[15] V. Nieto, V. Castro, F. Lopez, R. Ferro, and C. Gonzalez, "Model driven architecture software and interaction flow modelling language for tourism data acquisition in colombia," in *Communications in Computer and Information Science*, vol. 657, 2016, pp. 368–379.

[16] E. Umuhoza and M. Brambilla, "Model driven development approaches for mobile applications: A survey," in *Mobile Web and Intelligent Information Systems* (Lecture Notes in Computer Science), vol. 9847. Cham, Switzerland: Springer, 2016, pp. 93–107.

[17] Y. Rhazali, Y. Hadi, and A. Mouloudi, "A model transformation in MDA from CIM to PIM represented by Web models through SoaML and IFML," in *Proc. 4th Int. Colloq. Inf. Sci. Technol. (CiSt)*, Tangier, Morocco, Oct. 2016, pp. 116–121.

[18] N. Laaz and S. Mbarki, "Combining ontologies and IFML models regarding the GUIs of rich Internet applications," in *Proc. Int. Conf. Artif. Intell., Methodol., Syst., Appl.*, Aug. 2016, pp. 226–236.

[19] E. Umuhoza, H. Ed-douibi, M. Brambilla, J. Cabot, and J. Cabot, "Automatic code generation for cross-platform, multi-device mobile apps: Some reflections from an industrial experience," in *Proc. 3rd Int. Workshop Mobile Develop. Lifecycle*, Pittsburgh, PA, USA, Oct. 2015, pp. 37–44.

[20] A. Salini, I. Malavolta, and F. Rossi, "Leveraging web analytics for automatically generating mobile navigation models," in *Proc. Int. Conf. Mobile Services*, San Francisco, CA, USA, Jul. 2016, pp. 103–110.

[21] C. Bernaschina, S. Comai, and P. Fraternali, "IFMLEdit. org: Model driven rapid prototyping of mobile apps," in *Proc. 4th Int. Conf. Mobile Softw. Eng. Syst.*, Buenos Aires, Argentina, May 2017, pp. 207–208.

[22] C. Bernaschina, S. Comai, and P. Fraternali, "IFMLEdit.org: Model driven rapid prototyping of mobile apps," in *Proc. IEEE/ACM 4th Int. Conf. Mobile Softw. Eng. Syst. (MOBILESoft)*, Buenos Aires, Argentina, May 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7972808

[23] C. Bernaschina, S. Comai, and P. Fraternali, "Online model editing, simulation and code generation for web and mobile applications," in *Proc. IEEE/ACM 9th Int. Workshop Modelling Softw. Eng. (MiSE)*, Buenos Aires, Argentina, May 2017, pp. 33–39.

[24] R. J. Punnoose, R. C. Armstrong, M. H. Wong, and J. R. Mayo, "Survey of existing tools for formal verification," Sandia Nat. Lab., Livermore, CA, USA, Tech. Rep. SAND2014-20533, 551829, Dec. 2014.

[25] M. A. Basit-Ur-Rahim, F. Arif, and J. Ahmad, "Modeling of real-time embedded systems using SysML and its verification using UPPAAL and DiVinE," in *Proc. 5th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Beiing, China, Jun. 2014, pp. 132–136.

[26] J. Bowen and S. Reeves, "Generating obligations, assertions and tests from UI models," in *Proc. Hum.-Comput. Interact. (EICS)*, vol. 1, no. 1, p. 5, Jun. 2017.

[27] P. Gupta and P. Surve, "Model based approach to assist test case creation, execution, and maintenance for test automation," in *Proc. 1st Int. Workshop End-to-End Test Script Eng. (ETSE)*, Toronto, Ontario, Canada, Jul. 2011, pp. 1–7.

[28] S. L. D. Costa, V. V. G. Neto, and J. L. D. Oliveira, "A user interface stereotype to build web portals," in *Proc. 9th Latin Amer. Web Congr.*, Ouro Preto, Brazil, Oct. 2014, pp. 10–18.

[29] A. Amjad, F. Azam, M. W. Anwar, W. H. Butt, and M. Rashid, "Event-driven process chain for modeling and verification of business requirements—A systematic literature review," *IEEE Access*, vol. 6, pp. 9027–9048, 2018.

[30] C. Bernaschina, S. Comai, and P. Fraternali, "Formal semantics of OMG's interaction flow modeling language (IFML) for mobile and rich-client application model driven development," *J. Syst. Softw.*, vol. 137, pp. 239–260, Mar. 2018.
[31] R. Rodriguez-Echeverria, J. C. Preciado, J. Sierra, J. M. Conejero, and F. Sanchez-Figueroa, "AutoCRUD: Automatic generation of CRUD specifications in interaction flow modelling language," *Sci. Comput. Program.*, vol. 168, pp. 165–168, Dec. 2018.
[32] *Open Source MBUITC Generator Tool*. Accessed: Feb. 2018. [Online]. Available: http://ceme.nust.edu.pk/ISEGROUP/MBUITC/index.html

NAZISH YOUSAF received the B.S.E. degree from International Islamic University (IIUI), Islamabad, Pakistan, in 2015. She is currently pursuing the M.S. degree in software engineering with the Department of Computer and Software Engineering, CEME, National University of Sciences and Technology, Pakistan. Her research interests include model driven software engineering, web modeling, and interaction flow modeling languages.

FAROOQUE AZAM is currently an Adjunct Faculty Member of the Department of Computer and Software Engineering, College of Electrical Mechanical Engineering, National University of Sciences and Technology, Pakistan. He has been various software engineering courses, since 2007. His areas of interest include model driven software engineering, business modeling for web applications, and business process reengineering.

WASI HAIDER BUTT is currently an Assistant Professor with the College of Electrical and Mechanical Engineering, Department of Computer and Software Engineering, National University of Sciences and Technology, Pakistan. His areas of interest include model driven software engineering, web development, and requirement engineering.

MUHAMMAD WASEEM ANWAR is currently pursuing the Ph.D. degree with the Department of Computer and Software Engineering, CEME, National University of Sciences and Technology, Pakistan. He is a Senior Researcher and an Industry Practitioner in the field of model-based system engineering (MBSE) for embedded and control systems. His major research areas include model-based system engineering (MBSE) for complex and large systems.

MUHAMMAD RASHID received the bachelor's degree in electrical engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2000, the master's degree in embedded systems design from the University of Nice, Nice, France, in 2006, and the Ph.D. degree in embedded systems design from the University of Bretagne Occidentale, Brest, France, in 2009. He is currently an Assistant Professor with the Computer Engineering Department, Umm Al-Qura University, Mecca, Saudi Arabia.