

Automated Model-based UI Test Case Generation from Interaction Flow Modeling Language (IFML) Models



Author

Nazish Yousaf

FALL 2015-MS-15(CSE) 00000118918

MS-15 (CSE)

Supervisor

Dr. Farooque Azam

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

OCT, 2017

Automated Model-based UI Test Case Generation from
Interaction Flow Modeling Language (IFML) Models

Author

Nazish Yousaf

FALL 2015-MS-15(CSE) 00000118918

A thesis submitted in partial fulfillment of the requirements for the degree of
MS Software Engineering

Thesis Supervisor:

Dr. FarooqueAzam

Thesis Supervisor's Signature: _____

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,
ISLAMABAD
OCT, 2017

DECLARATION

I certify that this research work titled “*Automated Model-based UI Test Case Generation from Interaction Flow Modeling Language (IFML) Models*” is my own work under the supervision of Dr. FarooqueAzam. This work has not been presented elsewhere for assessment. The material that has been used from other sources has been properly acknowledged / referred.

Signature of Student

Nazish Yousaf

FALL 2015-MS-15(CSE) 00000118918

LANGUAGE CORRECTNESS CERTIFICATE

This thesis is free of typing, syntax, semantic, grammatical and spelling mistakes. Thesis is also according to the format given by the university for MS thesis work.

Signature of Student

Nazish Yousaf

FALL 2015-MS-15(CSE) 00000118918

Signature of Supervisor

COPYRIGHT STATEMENT

- Copyright in text of this thesis rests with the student author. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the author and lodged in the Library of NUST College of E&ME. Details may be obtained by the Librarian. This page must form part of any such copies made. Further copies (by any process) may not be made without the permission (in writing) of the author.
- The ownership of any intellectual property rights which may be described in this thesis is vested in NUST College of E&ME, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the College of E&ME, which will prescribe the terms and conditions of any such agreement.
- Further information on the conditions under which disclosures and exploitation may take place is available from the Library of NUST College of E&ME, Rawalpindi.

ACKNOWLEDGEMENTS

I am thankful to my Creator Allah Subhana-Watala to have guided me throughout this work at every step and for every new thought which You setup in my mind to improve it. Indeed, I could have done nothing without Your priceless help and guidance. Whosoever helped me throughout the course of my thesis, whether my parents or any other individual was Your will, so indeed none be worthy of praise but You.

I am profusely thankful to my beloved parents who raised me when I was not capable of walking and continued to support me throughout in every department of my life.

I would also like to express my gratitude to my supervisor **Dr. FarooqueAzam** and my co-supervisor **Dr. Wasi Haider Butt** for their constant motivation and help throughout this thesis. Also for Software Development and Architecture (SDA), Model-driven Software Engineering (MDSE) and Software Requirement Engineering (SRE) courses which they have taught me. I can safely say that I haven't learned any other engineering subject in such depth than the ones which he has taught.

I would also like to thank my Guidance Committee Members **Dr. Rashid Ahmed** and **Dr. Usman Akram** for being on my thesis guidance and evaluation committee. Their recommendations are very valued for improvement of the work. I would like to pay special thanks to **Muhammad Waseem Anwar** for his incredible cooperation. I appreciate his guidance throughout the whole thesis. I am also thankful to **Madeha Arif** and **Anam Amjad** for their support and cooperation.

Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

Dedicated to my exceptional parents whose tremendous support and cooperation led me to this wonderful accomplishment

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 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 of 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, a *model based testing approach* has been proposed for testing *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*. The main idea of this approach is to *provide 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. Proposed approach has been validated through multiple case studies.

Keywords: Model driven engineering (MDE), Web based application development, Model based testing, MBT, UI testing, GUI, Transition based testing, Navigation verification, Formal verification, UPPAAL

TABLE OF CONTENTS

DECLARATION	i
LANGUAGE CORRECTNESS CERTIFICATE	ii
COPYRIGHT STATEMENT	iii
ACKNOWLEDGEMENTS	iv
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	ix
LIST OF TABLES	x
CHAPTER 1: INTRODUCTION	12
1.1. Background Study	12
1.1.1. Interaction Flow Modeling Language (IFML).....	12
1.1.2. Model Based Testing (MBT).....	13
1.2. Problem Statement	14
1.3. Proposed Methodology	14
1.4. Research Contribution.....	15
1.5. Thesis Organization	16
CHAPTER 2: LITERATURE REVIEW	19
2.1. Review Protocol.....	19
2.1.1. Categories Definition.....	19
2.1.2. Selection and Rejection Criteria	20
2.1.3. Search Process	21
2.1.4. Quality Assessment	23
2.1.5. Data Extraction and Synthesis	25
2.2. Results and Analysis	26
2.2.1. Classification of significant researches.....	26
2.2.2. Model-driven web development approaches identification	27
2.2.3. Model-driven web development tools identification	29
2.2.4. Model-driven web development languages identification	31
2.2.5. Model-driven web approaches benefits and limitations.....	34
2.3. Research Gaps.....	37
CHAPTER 3: PROPOSED METHODOLOGY	42
3.1. Targeted IFML Constructs	42
3.2. Proposed Solution	44
CHAPTER 4: IMPLEMENTATION	48
4.1. Transformation Rules.....	48

4.1.1.	IFML to Test Case Transformation Rules	49
4.1.2.	IFML to State Transition Matrix Transformation Rules	50
4.1.3.	IFML to Source&Target Information Matrix Transformation Rules.....	50
4.1.4.	IFML to UPPAAL Transformation Rules	51
4.2.	Transformation Engine Architecture.....	53
CHAPTER 5: VALIDATION		58
5.1.	Online Auctions Case Study	58
5.1.1.	Requirement Specification.....	58
5.1.2.	Modeling.....	62
5.1.3.	Navigation Model and Test Case Generation	66
5.1.4.	Automated Navigation Verification.....	69
5.2.	Library Case Study.....	71
5.2.1.	Requirement Specification.....	71
5.2.2.	Modeling.....	71
5.2.3.	Navigation Model and Test Case Generation	73
5.2.4.	Automated Navigation Verification.....	74
CHAPTER 6: DISCUSSION AND LIMITATION		77
6.1.	Discussion	77
6.2.	Limitations	78
CHAPTER 7: CONCLUSION AND FUTURE WORK		80
APPENDIX A.....		81
REFERENCES		87

LIST OF FIGURES

Figure 1: MBT Process.....	14
Figure 2: Research Flow.....	15
Figure 3: Thesis Outline.....	17
Figure 4: Search Process.....	23
Figure 5: Selected researches per year.....	24
Figure 6: Selected researches per publisher.....	25
Figure 7: Example of IFML Model.....	43
Figure 8: Example of State Transition Matrix.....	45
Figure 9: Example of Source&Target Information Matrix.....	46
Figure 10: Main Interface of Model based UI Test Case Generator.....	48
Figure 11: Transformation Engine Architecture.....	53
Figure 12: IFML to Test Cases Transformation Engine.....	54
Figure 13: Online Auctions Domain Model.....	63
Figure 14: Online Auctions IFML Model (Diagram 1 of 5).....	64
Figure 15: Online Auctions IFML Model (Diagram 2 of 5).....	64
Figure 16: Online Auctions IFML Model (Diagram 3 of 5).....	65
Figure 17: Online Auctions IFML Model (Diagram 4 of 5).....	65
Figure 18: Online Auctions IFML Model (Diagram 5 of 5).....	66
Figure 19: Transformation for Online Auctions Model.....	67
Figure 20: State Transition Matrix.....	68
Figure 21: Source&Target Information Matrix.....	68
Figure 22: Test Case for Search Form.....	69
Figure 23: UPPAAL Model for Online Auctions.....	70
Figure 24: Deadlock and Reachability verification.....	70
Figure 25: Library Domain Model.....	72
Figure 26: Library IFML Model.....	72
Figure 27: Transformation for Library Model.....	73
Figure 28: Library State Transition Matrix.....	73
Figure 29: Library Source&Target Information Matrix.....	74
Figure 30: Test Case for Book Details.....	74
Figure 31: UPPAAL Model for Library System.....	75
Figure 32: Deadlock and Reachability verification for Library System.....	75

LIST OF TABLES

Table 1: Details of research works per database.....	21
Table 2: Details of search terms and search results	22
Table 3: Details of Data extraction and synthesis.....	25
Table 4: Results of Classification of selected researches.....	26
Table 5: Classification of Others category.....	27
Table 6: Identification of Model-driven Approaches for Web Development	28
Table 7: Identification of Tools for Model-driven Web Development	29
Table 8: Language Identification for Model-driven Web Development.....	31
Table 9: PIM level languages	32
Table 10: PSM level languages	32
Table 11: Transformation languages	33
Table 12: Comparative Analysis of MDWE Approaches.....	34
Table 13: Overall Comparative analysis.....	37
Table 14: Comparative analysis with model-based UI test case generation	38
Table 15: Transformation rules for IFML to Test cases	49
Table 16: Transformation rules for IFML to State Transition Matrix	50
Table 17: Transformation rules for IFML to Source&Target Information Matrix	50
Table 18: Transformation rules for IFML to UPPAAL model	52

Chapter 1

Introduction

CHAPTER 1: INTRODUCTION

This chapter offers a detailed introduction of the research. **Section 1.1** discusses the background study, **Section 1.2** presents the problem statement, **Section 1.3** gives proposed methodology in, research contribution is detailed in **Section 1.4** and **Section 1.5** contains thesis organization.

1.1. Background Study

The purpose of providing the background study is to introduce the main concepts used in this research. The concepts involved are; 1) Interaction Flow Modeling Language (IFML) and 2) Model Based Testing (MBT). The details of the following are given in subsequent sections.

1.1.1. Interaction Flow Modeling Language(IFML)

The Interaction Flow Modeling Language 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 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 software applications, defined in next section.

1.1.1.1. 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 us with 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 the 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.

1.1.1.2. 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 diagrams unlike UML which relies on multiple diagrams to express each concept.

1.1.2. Model Based Testing (MBT)

Testing is considered one of the most important and difficult phase of Software Development Life Cycle (SDLC). In the beginning, software products were used to be tested manually and testing was confined to only one phase in SDLC. As the software development methods evolved, testing was no longer done manually. Testers shifted from manual testing to scripted testing. But as the time passed, agile and other iterative software development methods were adopted, testing was no longer kept confined to one phase. Testing is now started right from the beginning of SDLC till the end. This is called Model-based Testing (MBT). MBT involves test case generation from software design models containing main functional aspects of system, in whole or in parts. MBT is also referred as Model-driven Testing (MDT). An overview of MBT process is shown in **Figure 1**. MBT provides the advantage of automated testing. Hence, there is no chance of human error and also it makes the development period shorter because of time efficient testing. Model driven Engineering (MDE) has been widely adopted in the recent years and many MBT tools are available in the market, some of which are MaTeLo[1][2], TestMaster[3], PyModel[4], JSXM[5], TEMA[6], UPPAAL[7][8], Conformiq[9], ParTeG[10], Tedeso[11] and Simulink[12] etc.

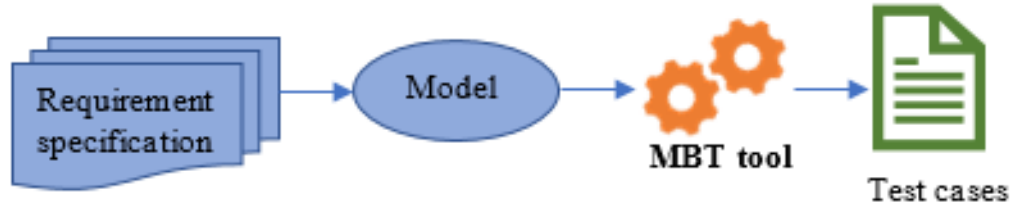


Figure 1: MBT Process

1.2. Problem Statement

Due to the growing complexity and size of web applications, manual testing of web interface becomes a time, resource and cost consuming process. Identifying web interfaces that can be used for testing such applications has become increasingly challenging. Moreover, changes in development phase cause changes in the front-end. (page layout can change, input elements are added or removed, data-flow of the pages is modified). These changes must be tested for error free application. Without the model driven approach, the tests are created manually, and the cost of testing is approximately 50% of the total development cost. Every change made to the interface must be synchronized with the tests i.e. when an element is added to a form, all functional tests associated with that form must be modified accordingly. So, in order to test our web application front-end in an effective and efficient manner, the test cases should be generated systematically through the IFML models at platform independent level without considering the development techniques. This leads to initiate the testing activity in the early requirement specification phases especially without considering the ultimate application development technologies.

1.3. Proposed Methodology

The entire research is done in a systematic way. Flow of the research is shown in **Figure 2**. First of all, we identify the problem, then we propose a solution to the identified problem. Then, we carry out a comprehensive systematic literature review which becomes the foundation of the proposed solution. Researches related to the proposed solution are analyzed and compared.

The proposed work includes a fully automated approach for obtaining the UI testing artifacts from IFML models at platform independent level i.e. in early phases of requirement and

analysis. The proposed tool provides facilities of modeling, transformation and verification. Mapping rules defined for the transformation become the basis of transformation engine. In the implementation phase, the transformation engine helps transform the IFML models into UI test case generation, UI testing matrices and UPPAAL navigation model. The navigation model generated by transformation engine can further be verified using verification feature of the tool and hence, providing the fully automated navigation verification. The proposed methodology has been validated for two case studies of different sizes.

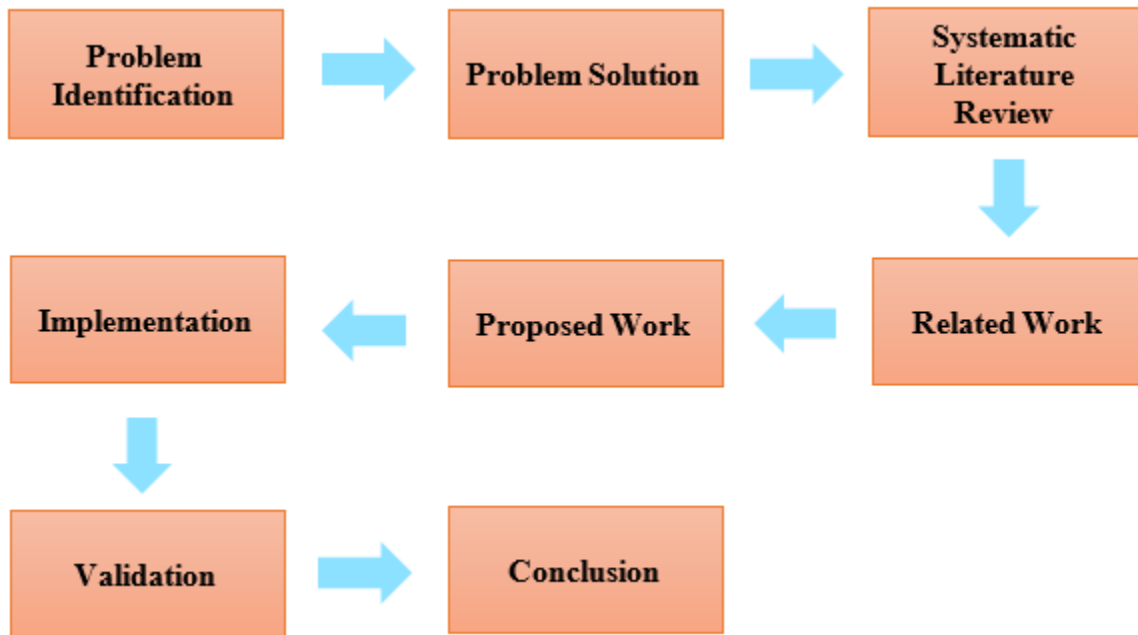


Figure 2: Research Flow

1.4. Research Contribution

Contributions made from this research work are two-fold i.e. comprehensive automated navigation verification and UI test case generation at platform independent level. Detailed set of contributions of the proposed approach are as follows:

- We have exploited Model Driven Engineering and Formal Verification technique for automated navigation verification. Timed Automata formalism has been used for this purpose.

- We have specifically developed a method to generate Timed Automata model from IFML model, so that it can be verified for reachability and deadlock freedom using UPPAAL model checker.
- We have introduced a model based approach in order to obtain testing artifacts. The main artifact produced using this approach is a complete UI test case document in-line with the specification document which is generated even before the development starts. The UI test case document covers exhaustive test cases for proper display of data on the web pages.
- Other testing artifacts i.e. transition matrices are also generated from which testers and developers can easily utilize other formalisms and testing techniques i.e. quick state transition testing.
- The transformation engine is developed using Java and Acceleo for the generation of full test case report (.txt) and also UPPAAL navigation model.
- We have provided validation of our proposed work using two benchmark case studies i.e. Online Auctions and Library case study.

1.5. Thesis Organization

Organization of the thesis is represented in **Figure 3**. **CHAPTER 1: INTRODUCTION** offers a brief introduction containing the background study, problem statement, research contribution and thesis organization. **CHAPTER 2: LITERATURE REVIEW** provides the detailed literature review highlighting the work done in the domain of web based application development using Model Driven Architecture (MDA). The systematic literature review is composed of four main sections. First section is review protocol which gives details on the methodology using which the literature review is carried out. Section two offers details on research works using MDA for web development. Whereas, section three highlights the research gaps that we encountered. **CHAPTER 3: PROPOSED METHODOLOGY** covers the details of proposed methodology used for identification of problem. **CHAPTER 4: IMPLEMENTATION** presents the detailed implementation regarding the proposed tool and transformation engine along-with its architecture. **CHAPTER 5: VALIDATION** provides the validation performed for our proposed methodology using two important case studies. The two case studies selected for validation purposes are of different domains and different sizes to make sure that our proposed approach works on every case. Case studies selected are Online Auction and Library case study. **CHAPTER 6: DISCUSSION AND**

LIMITATION contains a brief discussion on the work done and also contains the limitations to our research. **CHAPTER 7: CONCLUSION AND FUTURE WORK** concludes the research and recommends a future work for the research.

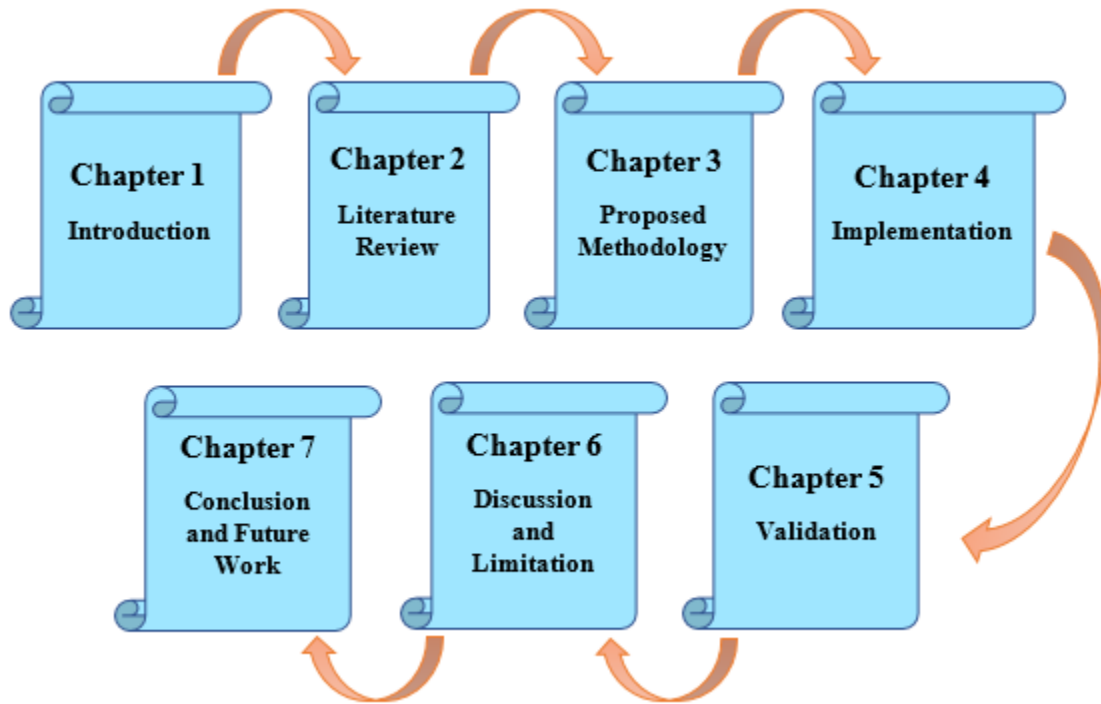


Figure 3: Thesis Outline

Chapter 2

Literature Review

CHAPTER 2: LITERATURE REVIEW

This chapter presents the literature review carried out for the research. **Section 2.1** discusses the review protocol, **Section 2.2** presents the results obtained from the review protocol and **Section 2.3** highlights the research gaps which form the foundation of our research.

2.1. Review Protocol

We carried out the review protocol development for our study, based on already defined Systematic Literature Review by Kitchenham[13]. This review protocol demonstrates the category definition, criteria of selection and rejection, assessment of quality, extraction of data and the mechanism used for data synthesis. The details of these elements are given in following sub-sections.

2.1.1. Categories Definition

We have categorized our researches into four main categories on the basis of different types of approaches used for model driven development of web applications. Later on, various approaches and tools used in these researches are identified. The research studies in these categories can overlap.

Profiling Category: This category contains the research works in which profiles either new or existing have been used for development of web applications. This category includes the research studies where either UML-extended profiles or user-interest profiling methods have been used.

Domain Specific Modeling Languages (DSML) Category: This category contains the research works in which domain specific modeling languages have been used for model based web development. WebML and Object-Oriented languages are placed in this category.

Service Oriented Architecture (SOA) Category: The research works in which service oriented architecture has been used along-with model driven approaches are to be placed under this category.

Others Category: This category includes the research studies that did not belong to any of the above-mentioned categories. It is divided into following three sub-categories.

- a. **Web user interface development:** This sub-category contains the research studies that do not lie in any of the above-mentioned three categories and focuses on web user interface development or web user behavior analytics using model driven approach. The research works on desktop, mobile web and rich internet applications(RIA) web interface are included in this sub-category. For example, [14]and[15] discussed the application of MDA for developing a model based technique that can be used to develop responsive web user interfaces for Rich Internet Applications (RIAs).[16]and [17]discussed the MDA integrated with Interaction Flow Modeling Language (IFML) for development and validation of web interfaces for mobile applications.
- b. **Business process modeling:** The research studies related to the MDA based business process modeling for web based applications are to be placed under this sub-category. Business Process Modeling Notation (BPMN) is used for modeling purposes.
- c. **Model based testing:** The research works in which model based testing of web applications has been discussed are to be included under this sub-category.

2.1.2. Selection and Rejection Criteria

The standard and benchmark for the inclusion and exclusion of this study are declared by using seven parameters. These factors defined to certify the validness of the responses of our questions. The studies that do not comply with and do not fulfill these seven parameters are not considered. Selection parameters for research works are given below,

1. **Subject relevance:** We selected only those papers which dealt with the model driven development of web applications. The selected work supports the responses of the research questions that we asked. Furthermore, we rejected unrelated research studies which did not include MDA.
2. **2010-2017:** We ensured the collection of latest studies by opting for those studies which lie in the years 2010 to 2017, and by not considering those studies which fall beyond the described period.
3. **Publishers:** Primarily five famous scientific databases were used, which are IEEE, ACM, ELSEVIER, SPRINGER and TAYLOR & FRANCIS; to ensure the inclusion of authentic and state of the art research works we opted for those papers which have been brought forward by the specified publishers. Details are given in **Table 1**.

4. **Imperative effects:** The papers that we chose have imperative positive impact on development of web-based applications. We tend to reject the studies that have no important subsequence on the development of web-based applications.
5. **Result-oriented:** The studies that we opted are result oriented, and their proposal and final conclusions are analyzed and proven by stable experimentation and factual data. Moreover, we do not consider those studies whose claims are proven by weak methodologies of validation.
6. **Redundancy:** We rejected redundant research studies and only most outstanding one of them was used.
7. **Proper Validation:** Selected researches must be validated by developing a web application using model driven development.

Table 1: Details of research works per database

Sr. #	Scientific Database	Type	Selected Research Works	No. of Researches
1.	IEEE	Journal	[18][19]	2
		Conference	[14][20][21][22][23][24]	6
2.	ACM	Journal	[15][25]	2
		Conference	[26][27][28][29]	4
3.	ELSEVIER	Journal	[30][31][32][33][34][35][36]	7
		Conference	[37][38][39]	3
4.	SPRINGER	Journal	[40][41][42]	3
		Conference	[16][17][43][44][45]	5
5.	TAYLOR & FRANCIS	Journal	[46][47][48]	3
Total				35

2.1.3. Search Process

The selection and rejection criteria depict that we have opted for five prime databases of publication (i.e. ACM, IEEE, Taylor & Francis, Springer and Elsevier) to perform the systematic literature review process. We used “2010–2017” year-filter on all the search terms to get the searches put out during 2010–2017, merely. Some of the search terms included (e.g. Model

driven web, Model based web development, MDA based web, Web modeling, WebML and IFML) as mentioned in **Table 2**. We used the “AND” and “OR” operators to accomplish the possible investigation outcomes necessary for our study. We followed the search process flow diagrams illustrated in **Figure 4**.

1. **Identification:** We specified multiple search expressions in five scientific databases and examined about **3509** results.
2. **Screening:** We excluded **2887** studies in the screening process because their title did not comply with our criteria.
3. **Eligibility:** We considered **622** researches and by accessing their full text and by reading their abstracts and results we discarded **483** researches because they did not match with our selection and rejection criteria. For example., [49]presented an article based on web languages but did not provide valid case study or website example so we rejected this study because it did not meet our eligibility criteria of validation mentioned in **Section2.1.2. Selection and Rejection Criteria.**
4. We performed a thorough qualitative and quantitative study of **52** researches by extracting their data and synthesizing it later for our research questions. After detailed examination of our **52** paperswe rejected **17** studies which did not fulfill our merit quantitative and qualitative criteria.
5. **Included:** We finally included remaining **35** papers because they fully comply with our set criteria for selection and rejection.
6. The details of selected research studies as per the publishers.

Table 2: Details of search terms and search results

Search Terms	Operators	Number of Search Results				
		IEEE	SPRINGER	ELSEVIER	ACM	T&F
Model driven web	AND	2	2	41	4	25
	OR	83	155	117	58	148
Model based web development	AND	2	10	20	0	0
	OR	113	294	344	144	53
MDA based web	AND	7	2	5	0	0
	OR	43	9	103	39	8
Web modeling	AND	11	2	35	1	6

	OR	1145	15	250	90	12
WebML	N/A	4	9	47	4	0
IFML	N/A	6	20	16	5	0

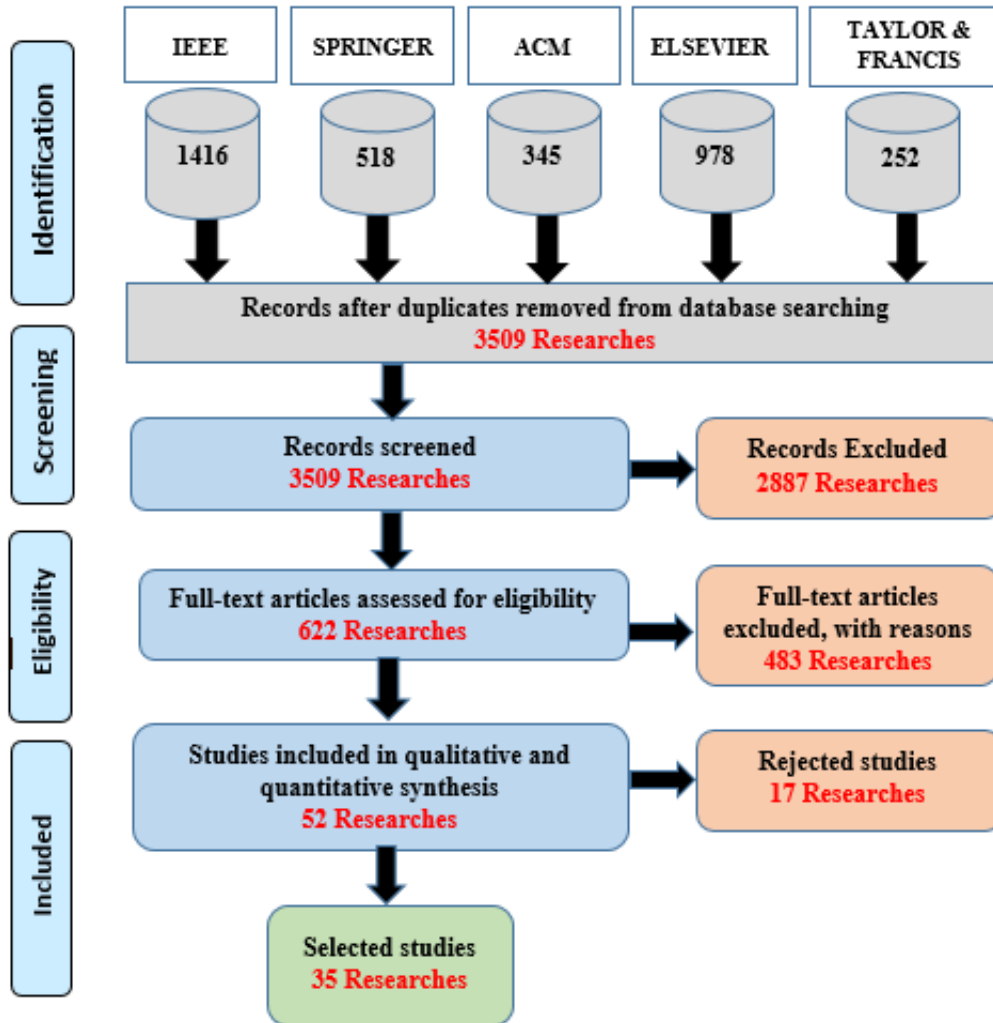


Figure 4: Search Process

2.1.4. Quality Assessment

We established the quality assessment criteria for understanding the importance of our result from the selected research studies. These criteria also help to define the trustworthiness of each research work we have selected and its fundamental discoveries:

1. The data evaluation of the researches is free from the ambiguous statements and relies on the solid facts and theoretical discerning.

2. Selected researches have been validated using appropriate validation techniques and approaches e.g. validation on some website or using case studies etc.
3. Tools information that has been used to perform different activities that helped us to validate our findings is provided.
4. As our intention is to examine or study the application of MDA for web development, model-transformation approaches and available tools for this purpose so, our goal is to include considerable total of most recent studies as much as possible. **Figure 5** shows the details of number of selected researches per year.
5. We have clearly and logically prepared and sorted the research by focusing on themes or ideas rather than the authors.
6. Uniqueness of the study is another important feature. Therefore, we have only included those research studies that are published in at least one of the following five well-known and internationally recognized scientific databases which are: ACM, SPRINGER, IEEE, ELSEVIER and TAYLOR & FRANCIS. Details given in **Figure 6**.
7. We have tried to avoid risk biasness at our best by taking researches of five of the most published databases i.e. IEEE, ACM, Elsevier, Springer and Taylor & Francis which helped us in sighting a significant amount of important research publications. There is a chance of risk biasness from many other sources for example, Wiley and Google Scholar etc., which we have not considered. This will not affect our research results in huge ratio but there would still be a difference of $\pm 10\%$.

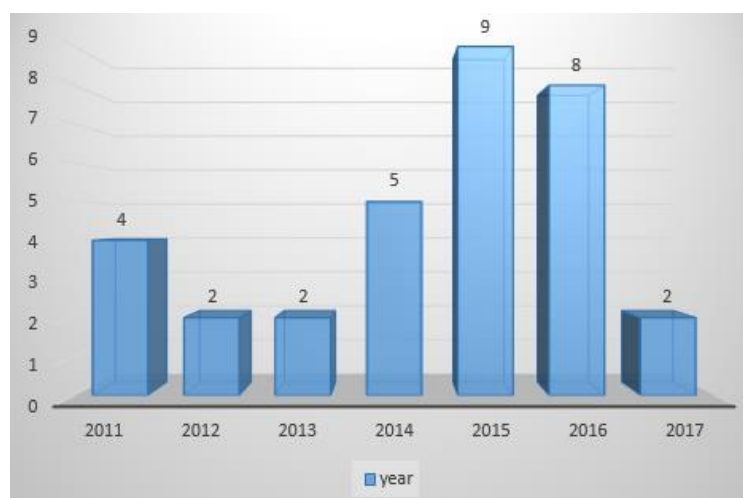


Figure 5: Selected researches per year

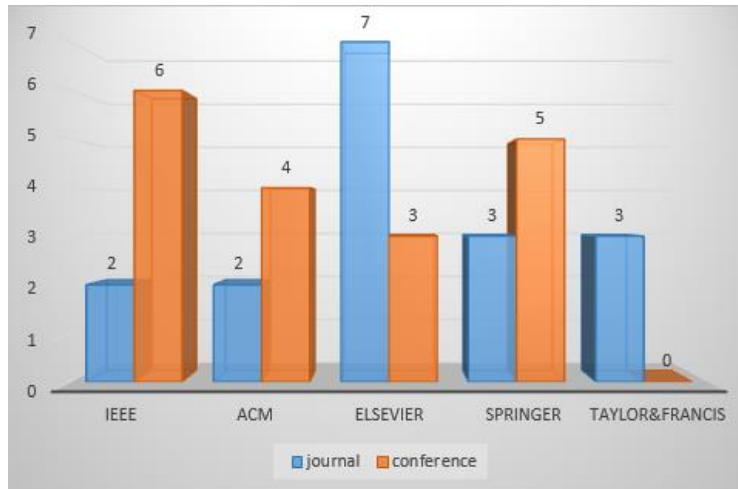


Figure 6: Selected researches per publisher

2.1.5. Data Extraction and Synthesis

Table 3 shows the data extraction and synthesis performed for our nominated researches to attain the answers of our research questions. After the data extraction, we conducted an inclusive analysis on model driven web based applications.

Table 3: Details of Data extraction and synthesis

Sr. #	Descriptions	Details
1.	Bibliographic information	Title of study, author, year of publication, details of publisher and the type of study (i.e. conference or journal publication)
Data Extraction		
2.	Overview	Basic idea of the selected research
3.	Results	Results obtained from the selected research
4.	Collection of Data (if any)	Qualitative or Quantitative
5.	Assumption(s)	For validating the results
6.	Validation	Methods used for validating the results of research idea.
Data Synthesis		
7.	Approach Utilization	Model driven approach used for web development
8.	Tools Identification	Tools used for model based web development
9.	Language Identification	Languages used for model based web development

Table 3 contains the details of data extraction and synthesis. We have defined some parameters, from serial number 2 to 6 for data extraction, from which we extracted the details of each selected research study to make sure that it conforms with our selection and rejection criteria. We have defined some parameters, from serial number 7 to 9 for data synthesis, considering these parameters we have performed detailed investigation of each selected research study. Each selected research study has been studied and investigated in detail in order to assign it to the equivalent category. Each selected research study has been studied intensively in order to extract the correct information regarding the approach utilization, tool and language identification as defined in serial number 7 to 9 respectively. We have tried to avoid risk biasness across individual studies by selecting the research studies that worked on complete and validated case studies for example, we have selected [16][29] and [45] because of the evidence of their methodology with complete IFML case studies.

2.2. Results and Analysis

2.2.1. Classification of significant researches

We have determined this Systematic Literature from 35 significant research studies and then we organized the selected researches into four pre-defined categories (**Section 2.1.1. Categories Definition**). This was done to acknowledge the relevant research works as shown in **Table 4**.

Table 4: Results of Classification of selected researches

Sr. #	Categories	No. of Researches	Research Identification
1	Profiling	7	[22][23][33][34][35][38][47]
2	DSML	15	[15][18][19][25][27][29][30][31][32][37][41][43][44][46][48]
3	SOA	4	[17][20][21][42]
4	Others	9	[14][16][24][26][28][36][39][40][45]

From the detailed investigation of 35 selected research studies we have analyzed that 7 research studies have been identified under profiling category. For example, [23] and [33] discussed a method of extending a new profile from UML for the introduction of web concepts at design level. [22] proposed a mechanism to enhance web application development by introducing MDA approach with a new user-interest profile. We also analyzed that Domain

Specific Modeling Languages (DSML) is the leading category in the model based software engineering practice. For example, [25] suggested different approaches like UML-based web engineering, Web Modeling Language, Object Oriented Web Solutions etc., that can be used for model driven development of web based applications. Therefore, we have selected 15 researches under DSML category.

On the other hand, only 4 research studies have been identified that are specific to Service Oriented Architecture (SOA-based) i.e., [21] and 9 research studies have been identified that do not correspond to any of the above-mentioned categories and hence are selected under the others category. Details of categories included in others category are given in **Table 5**.

Table 5: Classification of Others category

Sub-categories		No. of Researches	Research Identification
Web user interface development	Desktop web	3	[14][26][28]
	Mobile web	3	[16][36][45]
	RIA web	1	[24]
Business Process Modeling		1	[39]
Model based Testing		1	[40]

Table 5 enlists three sub-categories of others category. The sub-category of web user interface development has been divided into three parts. About 3 research works have been identified under the category of web user interface development for desktop applications, 3 research studies have been identified under the category of web user interface development for mobile applications and only one research study has been identified for Rich Internet Applications (RIAs) user interface development. Only one research work has been identified under the sub-category of business process modeling and one research work has been identified under the sub-category of model based testing.

2.2.2. Model-driven web development approaches identification

From the selected researches, we have identified a total of 16 approaches that have been used for model driven development of web applications. As per our research criteria, we only investigated the approaches in use since 2010 till now. **Table 6** summarizes widely used model based web development approaches now-a-days.

Table 6: Identification of Model-driven Approaches for Web Development

Sr. #	Model-driven Web Development Approaches	No. of Researches	Research Identification
1	UWE (UML-based Web Engineering)	7	[25][30][31][36][37][41][44]
2	WebML based	7	[19][25][29][30][31][32][44]
3	IFML based	9	[16][17][24][27][28][29][36][42][45]
4	WSDM (Web Site Design Method) using user-interest profiling	4	[19][22][23][25]
5	W2000 (UML Profiling)	6	[25][33][34][35][38][47]
6	OO-H, OOH4RIA and OOHDM (Object-Oriented Hypermedia Method)	4	[15][25][30][31]
7	RUX-Method	1	[31]
8	NDT (Navigational Development Techniques)	3	[25][31][44]
9	CSCS pattern for MOWS	1	[18]
10	WCF (web component framework)	1	[20]
11	SOD-M (Service Oriented Development Method)	1	[21]
12	VDM (Vienna Development Method)	1	[21]
13	FMrP (Function-Model-Responsive Presentation) Model	1	[14]
14	MockupDD (Mockup Driven Development)	1	[30]
15	BPMN (Business Process Model and Notation)	1	[39]
16	OOWS (Object Oriented Web Solutions)	1	[25]

From the detailed investigation of selected 35 researches, we have identified 16 approaches used for the model based web development purpose as enlisted in **Table 6**. Out of these 16 approaches, we have identified that UML-based Web Engineering, Object Oriented Hypermedia and Web Modeling Language based approaches have been used together in some of our researches such as in[25], [30] and [31]. Whereas, these approaches are used separately in some of the selected researches as well.

We have also identified that some approaches work on the idea of profiling which was selected as one of our categories for classification of the selected researches. For example, [33]and [34] discussed the approach called W2000 the idea of extending profiles form UML for the model based web application development purpose. Interaction Flow Modeling Language (IFML) has been identified by 9 selected researches as a widely-adopted approach for web interface development these days as discussed in [29] and [45]. 4 selected researches discussed another approach called Web Site Design Method (WSDM) which works on the idea of user-interest profiling.

On the other hand, there are some approaches such as Web Component Framework, MDA-SOA based approach, Vienna Development Method, Function-Model-Responsive-Presentation, Mockup Driven Development, Business Process Model & Notation and Object-Oriented Web Solutions which are only used in 1 research study.

2.2.3. Model-driven web development tools identification

From the selected researches, we have identified 15 tools that are being used for model driven development of web applications. As per our research criteria, we only investigated the tools in use since 2010 till now. **Table 7** gives a detail of tools used for model based web development classified according to the functionality they provide.

Table 7: Identification of Tools for Model-driven Web Development

Sr. #	Tools	Purpose			No. of Researches	Research Identification
		Modeling	Model Transformation			
			M2T	M2M		
Framework (5)						
1	Eclipse Kepler with	√			4	[20][15][33][47]

	Papyrus Project]
2	Eclipse Kepler with Acceleo Project		√		2	[20][33]
3	OO-Method by INTEGRANOVA		√	√	1	[35]
4	NDT Suite		√	√	2	[25][44]
5	Eclipse GMF	√			3	[18][25][37]
6	Eclipse EMF		√	√	4	[14][23][34][38]
Modeling (1)						
1	Eclipse Ecore (MOF)	√			3	[25][33][41]
Transformation (3)						
1	KM3			√	1	[41]
2	MedniQVT	√		√	4	[21][22][23][38][47]
3	WebDSL generators using Stratego/XT		√	√	1	[41]
Modeling & Transformation (6)						
1	MagicDraw	√		√	1	[23]
2	WebRatio and WebRatio BPM	√	√		6	[19][27][29][30][32][45]
3	ArgoUWE and MagicUWE	√		√	1	[25]
4	Mockup-to-HTML Tool	√		√	1	[30]
5	Interactive Tagging Tool	√		√	1	[30]
6	Demo Sandbox Environment	√		√	1	[30]

Table 7 shows the details of tools identified for the model driven development of web based applications. These tools are classified under four categories depending upon the functionality of each. Table also shows the features or purpose of each identified tool i.e., modeling and transformation. In transformation, we have further divided it into two categories, model-to-text or code(M2T) and model-to-model(M2M). From the detailed study of 35 selected researches we have identified 5 tools under the framework category where Eclipse Kepler was found an efficient framework with different plugins available for different purposes[20]. Only one tool i.e., Eclipse Ecore(MOF) has been identified under modeling category. Under transformation category, 3 tools have been identified. On the contrary, 6 tools have been identified which provide both modeling and transformation functionality. From the above table, it is easily distinguishable that both types of transformations are only supported by INTEGRANOVA, NDT Suite, Eclipse EMF and WebDSL generators.

2.2.4. Model-driven web development languages identification

We have divided the languages used for model driven web development into three categories on the basis of their purpose. We have identified a total of 23 languages that are used for model based development of web applications. **Table 8** defines the three categories we have developed for languages used in our selected research works. **Table 9** shows that PIM (Platform Independent Model) level languages are further divided into two categories, GPMLs (General Purpose Modeling Languages) and DSLs (Domain Specific Languages) and **Table 10** gives the information regarding PSM (Platform Specific Model) level languages. **Table 11** summarizes transformation languages and also specifies whether the language is used for Model to model, model to text transformation or both.

Table 8: Language Identification for Model-driven Web Development

Sr. #	Category	No. of Languages included	Research Identification
1	PIM level languages	9	[15][16][17][19][20][21][22][23][24][25][27][28][29][30][31][32][33][34][35][36][38][39][41][42][44][45][48]
2	PSM level	3	[15][18][20][21][22][23][24][33]

	languages		
3	Transformation languages	11	[21][22][23][38][39][41][47]

From the detailed analysis of 35 selected researches, we have classified the languages used in them in three categories which are the building block of any model driven engineering process. We have selected 9 languages under the category of PIM level languages which are basically general modeling languages as shown in **Table 9**. Whereas, **Table 10** shows that 3 languages have been identified under the category of PSM level languages. For transformation of Platform independent model to a platform specific model, some transformation languages are used, 11 of them have been identified by the examination of our selected researches. Details given in **Table 11**.

Table 9: PIM level languages

Sr. #	Category	Language	No. of Researches	Research Identification
1	GPMLs	UML	11	[15][20][21][23][31][32][33][34][36][38][39]
		IFML	9	[16][17][24][27][28][29][36][42][45]
2	DSLs	WebML	11	[15][19][21][29][30][31][32][41][43][44][48]
		MobML	1	[27]
		UWE	6	[22][25][30][41][43][44]
		MockupToME	1	[34]
		MIDAS	1	[41]
		OOWS	2	[35][41]
		Netlison	1	[41]

Table 10: PSM level languages

Sr. #	Languages	No. of Researches	Research Identification
1	XML	5	[15][18][20][23][33]
2	WSDL (Web Service Description Language)	2	[21][22]
3	WS-BPEL (Web Service Business Process Execution Language)	1	[21]

Table 11: Transformation languages

Sr. #	Languages	No. of Researches	Research Identification
(M2M)			
1	QVT and QVT-R	7	[21][22][23][24][38][41][47]
2	ATL (Atlas Transformation Language)	4	[21][39][41][47]
(M2T)			
1	AndroMDA	2	[24][41]
2	Stratego/XT (grammar)	1	[41]
3	XPand	1	[41]
4	Velocity	1	[41]
5	Groovy scripting language	1	[41]
(M2M + M2T)			
1	TCS (DSL for textual concrete syntaxes)	1	[41]
2	AspectJ	1	[38]
3	openArchitecture-Ware's xTend	1	[41]
4	openArchitecture-Ware's Workflow language	1	[41]

Platform Independent level languages are further categorized into two categories as enlisted in **Table 9**. First category is GMPLs, which includes general purpose modeling languages used for software development or software interface development such as UML and IFML. UML is widely used PIM level language, as suggested in 11 of our selected researches. 9 research studies have been identified in which IFML has been used. IFML is a relatively new approach identified by 9 research studies and has been adopted for the interface development as discussed in [42] and [45]. On the other hand, DSL is the second category which includes the domain specific languages such as WebML, UWE, OOWS etc. WebML is a language used for the web domain specifically as discussed in detail in 11 of the selected researches. [27] suggested

a new approach for mobile application development, language extended from IFML for this purpose has been named as MobML.

Table 10 shows the details of platform specific level languages that have been identified from our selected research studies. 5 research studies focused mainly on eXtensible Markup Language (XML) which was designed to store and transport data and is machine and human readable. Web Service Description Language (WSDL) and Web Service Business Process Execution Language (WS-BPEL) have been identified from 2 and 1 research study respectively.

Table 11 gives the detail of transformation languages identified from the detailed study of our selected researches according to the transformation supported by them. Query/View/Transformation (QVT) is the model-to-model transformation language standardized by Object Management Group for model transformation in the context of MDA. 7 research studies have been identified focusing on QVT and QVT-R(Relational). As suggested by [23] QVT-R can be used for transformation of system functionality from PIM to PSM level. ATLAS Transformation Language is another language used for the same purpose as identified by 4 of our selected researches. As per the category of model-to-text transformation languages, by our analysis we have identified that AndroMDA is a strong M2T transformation language. Other less used languages identified in the same context are XPand, Stratego/XT, Velocity and Groovy as suggested by [41]. Whereas, some less used transformation languages in the context of model driven web based application development which provide both model-to-model and model-to-text transformation such as TCS, Velocity, openArchitecture-Ware’s xTend and Workflow have been identified by only one research work [41].

2.2.5. Model-driven web approaches benefits and limitations

From an unbiased point of view, there are some clear benefits and limitations that can be identified from the 16 model driven approaches identified for development of web based applications. **Table 12** summarizes some of the important differences in context of web application 3 tier architecture supported by these MDA based approaches identified already in **Table 6**, as well as the notations supported and tool support available for them.

Table 12: Comparative Analysis of MDWE Approaches

MDWE Approaches	Supported Tiers	Supported	Tool Support
-----------------	-----------------	-----------	--------------

	Presentation	Business Logic	Data	Notation	
UWE	√	√		UML	AngroUWE, MagicUWE
WebML based	√	√	√	E-R, UML	WebRatio
IFML based	√	√	√	E-R, UML	WebRatio Eclipse IFML plugin
WSDM using user-interest profiling	√	√	√	UML	N/A
W2000	√	√	√	HDM, UML	N/A
OO-H, OOH4RIA and OOHDM	√	√	√	E-R, UML	CASE tool
RUX-Method	√	√	√	UiML	RUX tool
NDT	√	√		UML, OCL	NDT Suite
CSCS pattern for MOWS		√		UML	N/A
WCF	√	√		UML	N/A
SOD-M	√	√	√	E-R, UML	SOAP toolkit
VDM	√	√		UML	VDM toolkit
FMrP Model	√			E-R, UML	N/A
MockupDD	√	√	√	UML	Mockup-to-HTML
BPMN	√	√	√	UML	WebRatio BPM
OOWS	√	√	√	UML	OO-Method

All the above approaches also support their own notations.

Table 12 shows the detail of each tool's supported features from the 3-tier web architecture. These three tiers i.e., presentation, business logic and data in web application are based on separation of concerns. Presentation tier is user interface which is the top most level of a web application. Interface shows the interaction between user and the application. This tier is supported by all of the model based web development approaches identified by our research.

Second tier is the business logic tier which deals with the data processing and makes logical decisions of the application. Third and last tier is the data or persistence tier which stores the information and retrieves it from a database. This information is passed on to the business logic tier for processing and then eventually is sent back to the user at presentation layer. We have analyzed that the persistence layer is not supported by many approaches yet, which is one of the major limitations to these approaches.

Another important observation made from **Table 12** is that most of the identified approaches use models which are based on UML notations, which is commonly used standard in numerous varieties of software modeling. UWE (UML-based Web Engineering) and WSDM (Web Site Design Method) use UML extensions based on profiling with specific notations and stereotypes. WSDM is based on user-interest profiling an idea of MDA, it tries to decouple the functionality of systems defined at PIM level from the platform on which it runs at PSM level, to preserve the system functionality even if changes have been done in the underlying technology platform. These methods are complete, consistent and reliable when we consider modeling the client and server pages in a web based application. W2000 is also an extension and customization of UML with HDM (Hypermedia Design Model) based concepts of web design. Web Component Framework (WCF) at design level uses profiles extending UML with the introduction of web domain concepts. Later, the WCF framework which follows component based methodology is extended to support the development of web application. WCF does not have the tools solely used for its support but Eclipse Kepler with UML development and code generation plugins such as Papyrus and Acceleo Projects can be used. Model driven and Service Oriented approaches are also based on UML extension to get a domain specific language at PIM level. These two approaches are often combined to get an approach called SOD-M (Service Oriented Development Method) for the development of web application interfaces. Where most of the approaches are extending UML, RUX method is a rich user experience model used for web applications works on the concept of UiML (User interface Modelling Language). CSCS is a Configuration, Setting and Current State pattern used for Model Oriented Web Services (MOWS) supported by Eclipse GMF. WebML has been observed to be the most active approach used for web based application modeling. WebRatio environment provides full support for WebML and IFML whereas, WebRatio BPM provides support for large business process applications. Notation used for this purpose is again UML, in Business Process Model and

Notation (BDMN), an approach that allows models describing the business to move towards models presenting design and analysis of software product.

2.3. Research Gaps

This section discusses the research gaps and the proposed solution. On whole, we have identified six research works related to model based UI test case generation. **Table 13** 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 the reference number of the selected work, *2) Publication Year* is used to represent the year of publication of the selected research, *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.

Table 13: Overall Comparative analysis

Reference #	Publication Year	Modeling Language	Tool Support	Testing Aspects
[40]	2014	UML	None	Navigation
[51][52]	2015	IFML	Partial	Functional
[53]	2015	UML	None	Navigation
[54][55][56]	2017	IFML	Complete	Navigation
[59]	2017	UML	Partial	Functional
[60]	2011	BPMN	Partial	Functional
Our proposed approach	2017	IFML	Complete	UI Navigation

In **Table 13**, 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, three 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 language for representation of GUI. For example, Vikas S. et al., [40] 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. Karel Frajtek et al., [51][52] worked on generation of executable test cases in JavaScript format using MDE transformations and a template engine. The tests are executed using Jasmine test runner but no verification has been provided in the research paper. Eman M. Saleh el a., in [53] 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 Carlo Bernaschina[54] 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. Judy Bowen et al., [59] 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. Priya Gupta et al., [60] 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 14 presents the comparative analysis of our proposed work with the researches which worked on model based UI test case generation. We have selected five parameters for comparison; *1) Reference #* is used to represent the 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. Yes and No indicate the presence and absence of the parameter, respectively.

Table 14: Comparative analysis with model-based UI test case generation

Reference #	Automated Test Cases	Navigation Testing	Automation level	Formalism	Applicability
[28][29]	Yes	No	Semi-automated	None	Narrow
[54][55][56]	No	Yes	Fully-automated	Petri nets	Narrow
Our proposed approach	Yes	Yes	Fully-automated	Timed Automata	Broad

Table 14 provides a summary of overall studies found on Model based GUI testing along with the proposed approach. Karel Frajtak et al., [28][29] 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. Carlo Bernaschina et al., [54] 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. Extended work of Carlo Bernaschina et al., is presented in [57] and [58].

Although Carlo et al., presented a very good work which illustrated the strength of IFML based modeling for web application code generation, but we have identified some of the problems after using IFML.Edit.org. 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 us complete UI testing including the navigation testing and the automated test cases 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.

Chapter 3

Proposed Methodology

CHAPTER 3: PROPOSED METHODOLOGY

This chapter contains details of the proposed methodology. **Section 3.1** discusses the targeted IFML constructs and **Section 3.2** provides detailed proposed solution.

3.1. Targeted IFML Constructs

IFML metamodel provides the semantics and structure of constructs used in IFML. UML profile in IFML metamodel defines the syntax used to express IFML models in UML. IFML metamodel 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. We have only targeted the core concepts and some of main extensions concepts. A brief description of some of the core concepts of IFML 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, **Figure 7** shows a simple IFML model from movies case study. It explains simple scenario that if the user wants to add a 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. **Figure 7** 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 **Figure 7**, 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 **Figure 7**.
- <<Action>> Actions are triggered as an effect of events and are executed before change of state of the interface. Actions can contain usual functions i.e. UPDATE, ADD or DELETE.

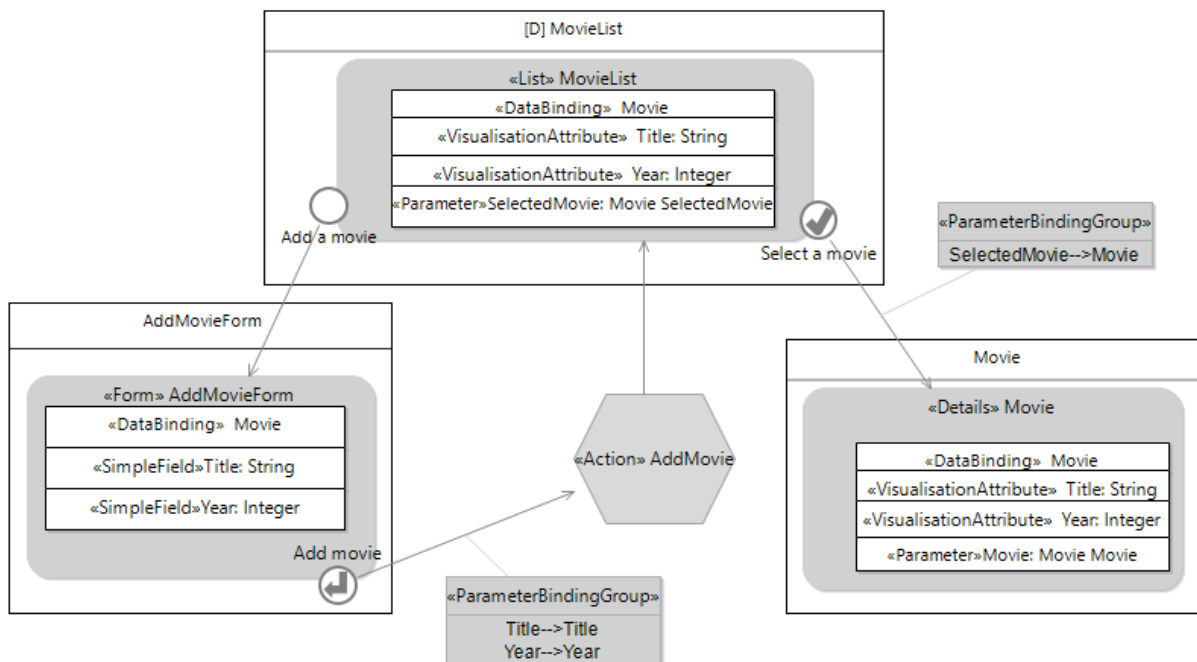


Figure 7: Example of IFML Model

3.2. Proposed Solution

We have proposed an approach based on Model Driven Software Engineering and formal verification. MDSE involves transformations for obtaining the code or target model from source model. Two types of transformations can be performed in MDSE. First type of transformation is Model to Model (M2M) transformation in which a target model is obtained from a source model. Second type of transformation is called Model to Text (M2T) transformation in which code or text is generated from target model. The source models can be of any design type i.e. UML model and IFML core model. Our approach used UML model for domain modeling and an IFML model for UI design of the web application. The approach takes both these models as input and applies M2T transformations in order to obtain desired outcomes. The transformations result in automated navigation verification testing and multiple testing artifacts.

In order to obtain the complete test case document for the web application testing, a M2T transformation is applied on UML and IFML model. Each view components of type Form, List and Details is mapped to single test case in the generated test case document. Simple fields and selection fields from Forms in the IFML model are used to obtain input elements along-with the required input type for the test cases. Events on the forms are considered as the final submission step in the test cases. Data binding from the lists in IFML model are exploited in order to obtain the domain element name and Visualization attributes from lists give the attributes of the domain element to be displayed in list on a particular web page. This domain element is usually a class or entity in the UML domain model of the web application. Similarly, test cases for Details view component of the IFML model are retrieved.

Other testing artifacts generated from M2T transformation on IFML model are State Transition matrix and an information matrix containing navigational details. The detailed navigational matrix has been referred as Source&Target Information matrix. The State Transition matrix is a matrix containing states as its first row and first column. Each column is checked against all the elements of first row. If a transition exists between two states, then particular cell of the matrix is marked 'T'(true) otherwise 'F' (false). An example of such matrix for a simple switch operation is shown in **Figure 8**. Two states 'ON' and 'OFF' are represented in first row and first column of the matrix and after checking the transition between two states, T and F values are filled. In order to obtain this matrix a transformation is applied on IFML model

and view containers from the model are selected to represent states in the matrix. Checking the transition in IFML model is not simple, Navigation flow is checked for multiple possible combinations i.e. from one container to the other, from component of one container to other container, from component of one container to the component of other container, from event of one container to other container, from event of one container to the component of other container, from event on component of one container to other container, from event on component of one container to the component of other container etc. and T/F values are assigned to respective cells of the matrix. This matrix is used as a black box testing technique and is useful for quick state transition testing of the web application where states are taken as web pages and navigation between them is checked. It helps testers to expose the invalid or unintended states.

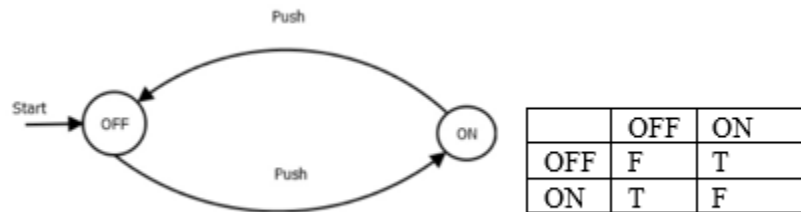


Figure 8: Example of State Transition Matrix

Another testing artifact called Source&Target information matrix is also generated from the transformation applied on IFML model. This matrix contains the details about the source and target pages in case of a valid navigation. The matrix covers detailed attributes of source page and also contains the navigational data or parameters to be passed from source page to the target page. An example of such matrix for the movies IFML model (**Figure 7**) is shown in **Figure 9**. The transformation includes the view containers along-with their landmark, default and XOR attributes and the parameter binding groups in order to retrieve the required cells of our matrix. This matrix helps developers to include primary parameters or arguments to be passed in order to carry out successful navigation. Each row of Source&Target Information Matrix contains test case for each navigation in the IFML model eventually helping the testers too.

Source Page	Landmark	Default	XOR	Parameter Binding	Target Page
MovieList	False	True	False	null	AddMovieForm
MovieList	False	True	False	movie	Movie

Figure 9: Example of Source&Target Information Matrix

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 the transitions between them are also represented using the checks used to obtain the State Transition matrix. The resulting model can be opened in UPPAAL model checker tool in order to verify the reachability and deadlock properties; hence, providing fully automated navigation verification.

Chapter 4

Implementation

CHAPTER 4: IMPLEMENTATION

This chapter presents the implementation details for Model Based UI Test Case Generator (MBUITC) (a tool we have developed for automated model-based UI testing and navigation verification for IFML models). The tool we have implemented has three main features. 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 UPPAAL. Main interface of MBUITC is shown in **Figure 10**. MDE has become the basis of our work. **Section 4.1** presents the transformation rules used to develop MBUITC and **Section 4.2** discusses the architecture of transformation engine.

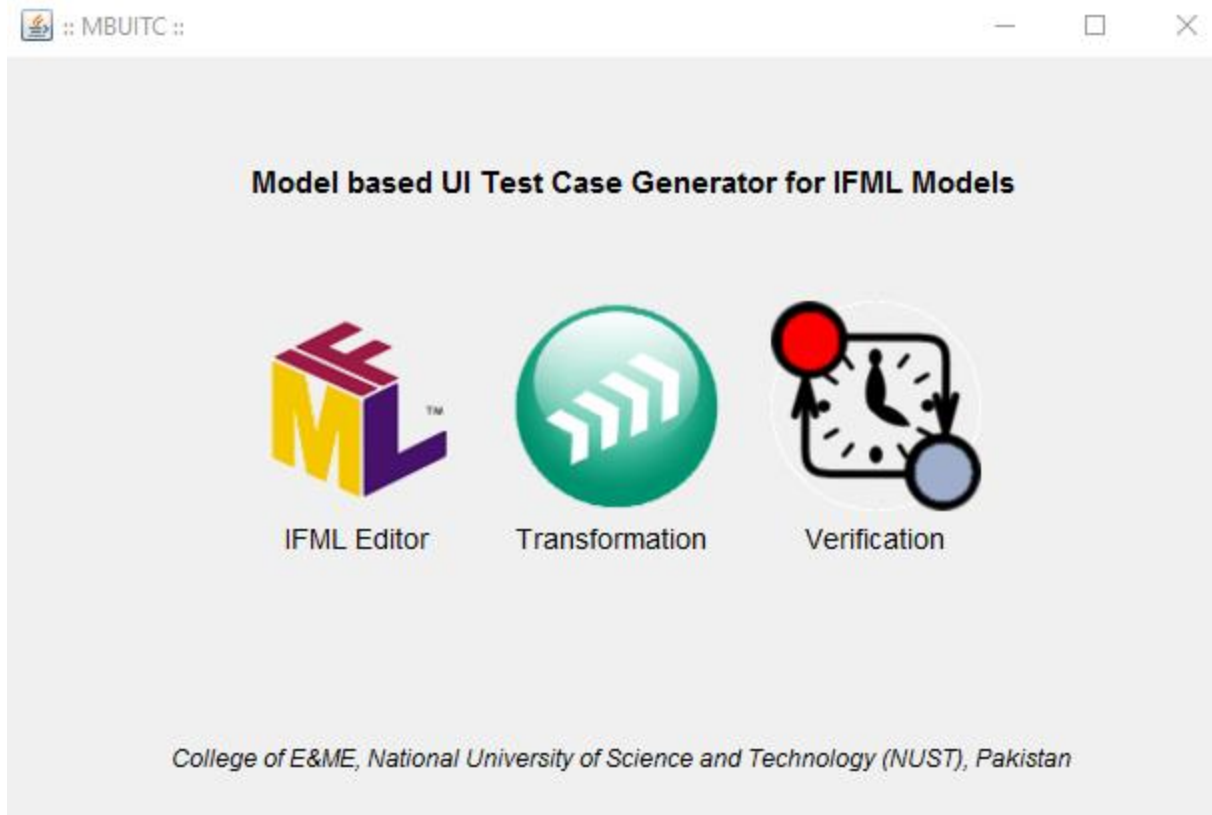


Figure 10: Main Interface of Model based UI Test Case Generator

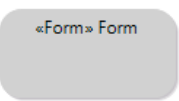
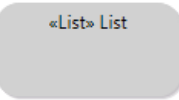
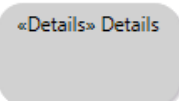
4.1. Transformation Rules

In this section, we have described the transformation rules in detail, which we have defined in order to transform the IFML models to testing artifacts.

4.1.1. IFML to Test Case Transformation Rules

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 15** 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 did not capture the UI details.

Table 15: Transformation rules for IFML to Test cases

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

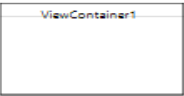

IFML *ViewComponent* of type *Form* is mapped to its respective test case in the test case document. Name of *Form* is mapped to test case name. *SimpleField* and *SelectionField* of each form is mapped to check for the type of input value and selected value, respectively. *onSubmit* event on the form 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.

4.1.2. IFML to State Transition Matrix Transformation Rules

In this section, we have explained the mapping rules used for our transformation of IFML core model to the State Transition Matrix. The transformation rules defined in **Table 16** are used to transform the concepts from IFML model to the State Transition Matrix.

Table 16: Transformation rules for IFML to State Transition Matrix

IFML Model	State Transition Matrix	Description
ViewContainer 	State	<i>ViewComponent</i> --name→ <i>First</i> row and column
NavigationFlow 	Transition	<i>NavigationFlow</i> →”True” and “F” values.

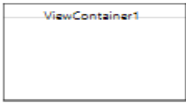

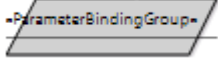
ViewComponent in IFML core model is transformed into state in the State Transition matrix. Each *viewContainer* name is mapped to elements of first row and column of the State Transition matrix. *NavigationFlow* in IFML model is mapped to transition in the State Transition matrix where it is represented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

4.1.3. IFML to Source&Target Information Matrix Transformation Rules

In this section, we have explained the mapping rules used for our transformation of IFML core model to the Source&Target Information Matrix containing the detailed test cases for each navigation in the IFML model. The transformation rules defined in **Table 17** are used to transform the concepts from IFML model to the Source&Target Information Matrix.

Table 17: Transformation rules for IFML to Source&Target Information Matrix

IFML Model	State Transition	Description

	Matrix	
ViewContainer 	Source/Target name	<i>ViewContainer</i> → Source and target states. <i>ViewContainer</i> --(isLandmark=true) → true value <i>ViewContainer</i> --(isDefault=true) → true value <i>ViewContainer</i> --(isXOR=true) → true value
NavigationFlow 	Navigation	NavigationFlow → Navigation
ParameterBindingGroup 	Parameter	<i>ParameterBindingGroup</i> -- <i>ParameterBinding</i> → parameter value

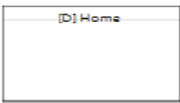



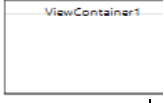

ViewContainer in IFML core model is mapped with source and target in the Source&Target Information Matrix. One container can act as source in one case and target in another. “isLandmark”, “isDefault” and “isXOR” attributes of each *ViewContainer* as “true” and “false” values are mapped to their respective 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*.

4.1.4. IFML to UPPAAL Transformation Rules

Mapping rules used for our 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 has been used for verification of resulting navigation model. UPPAAL provides simulation and verification of the models in which timed automata has been used. UPPAAL model is comprised of states and transitions[60][61]. *location* in UPPAAL model represents a state and *initiallocation* is used to represent initial state of the model. *Edge* in UPPAAL model is used to represent transition. For our approach, we do not need to include the time and guard constraint in UPPAAL model because only a navigation model is required for verification of reachability and

deadlock in our web navigation, so only states and transitions 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 18** along-with the graphical notations of source and target transformation.

Table 18: Transformation rules for IFML to UPPAAL model

IFML Model	UPPAAL Model	Description
Home ViewContainer 	Initial location 	$ViewContainer \text{--} Home(isDefault=true) \rightarrow initiallocation$
NavigationFlow 	Edge 	$NavigationFlow \rightarrow Edge$
ViewCont ainer 	Locat ion 	$ViewContainer \rightarrow location$

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 UPPAAL model. *Initiallocation* is used to represent the initial state of the system. *NavigationFlow* in the IFML model is mapped to *Edge* in the UPPAAL model. In order to create an Edge between two locations in UPPAAL 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 UPPAAL 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. We have described

the details of applicability of these rules in **CHAPTER 5:VALIDATION**. Two case studies have been used to verify the transformation rules mentioned above.

4.2. Transformation Engine Architecture

Architecture of our transformation engine is described in detail in **Figure 11**. We have implemented a transformation engine based on model based test case idea. This transformation engine fully automates the testing phase of SDLC by providing automated test cases from IFML models. Tool used for transformation of UML and IFML models to test cases and navigation model is Eclipse Acceleo. Our transformation engine is composed of three main components which are Main Interface, Acceleo Transformation and Java Services. Details of functionality performed by each component is explained below.

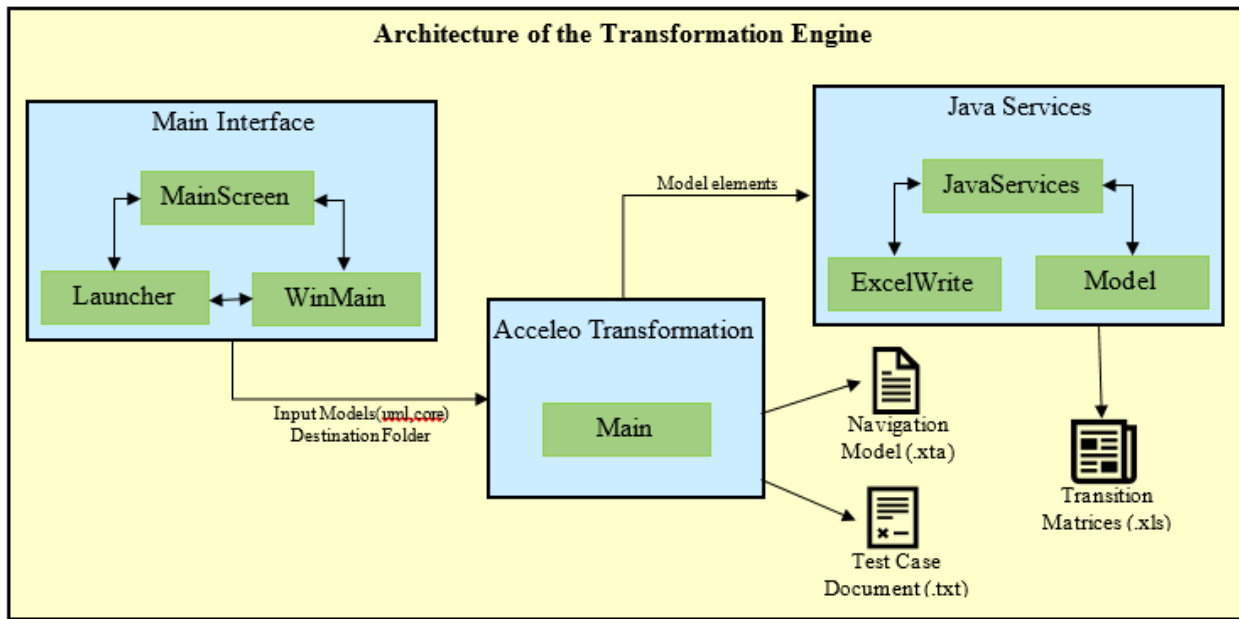


Figure 11: Transformation Engine Architecture

Main Interface: Main Interface component consists of three classes i.e. *MainScreen*, *Launcher* and *WinMain*. These three classes are used in development of graphical user interface of our tool. *MainScreen* is a class which provides execution and *WinMain* and *Launcher* contain its actual functionality. When the main screen of tool is opened (**Figure 10**), it provides us with three options i.e. IFML Editor, Transformation Engine and UPPAAL model verification. IFML Editor and UPPAAL can be opened directly whereas the interface of our transformation engine is shown in **Figure 12**. 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 generated the output files he needs. 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 [61]. To represent if the transformation has been performed successfully or if some error has occurred while transformation, a *Status* bar is shown. *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.

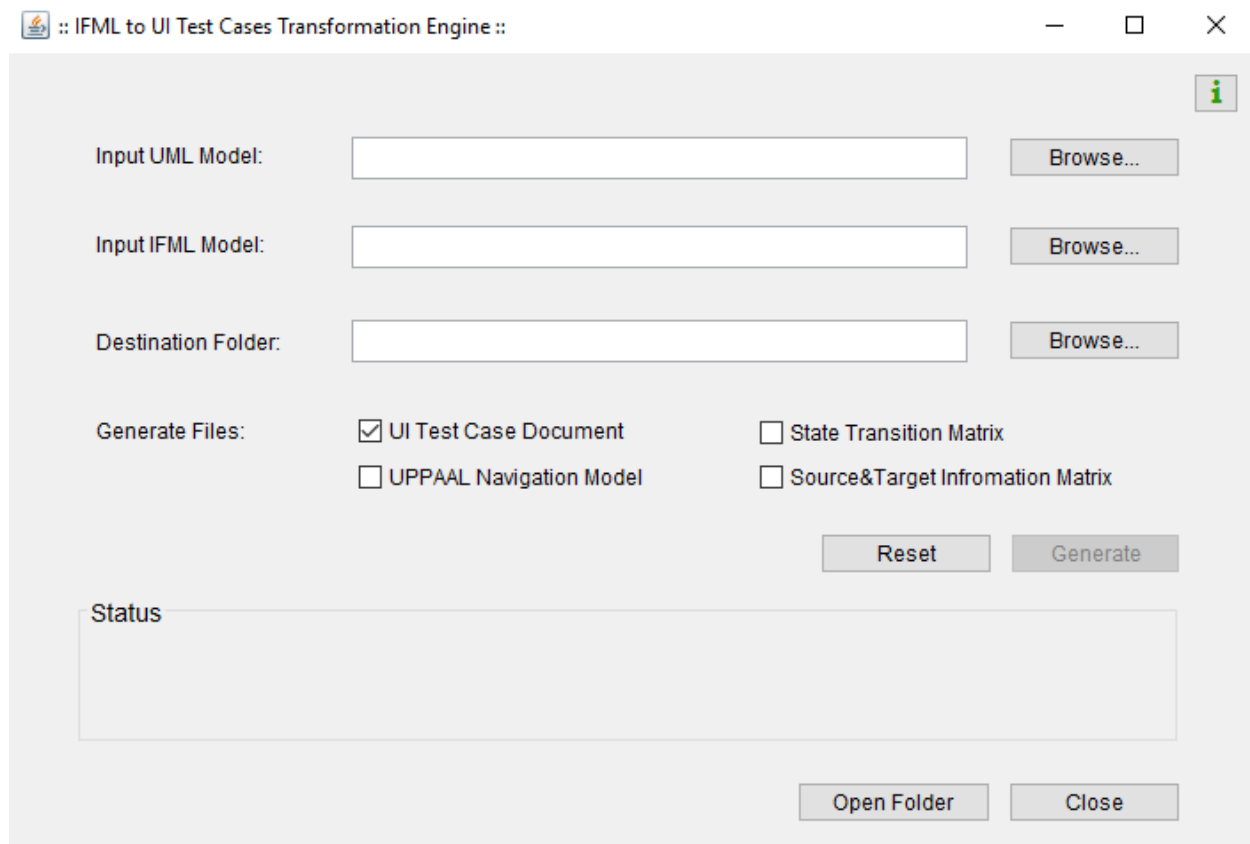


Figure 12: IFML to Test Cases Transformation Engine

Accelo Transformation: Main interface takes UML and IFML models as input and passes them to Accelo Transformation. Foremost files included in Accelo Transformation are *Main.java* containing java code for transformation and *main.mtl* containing accelo 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 3.1**. *Main.java* and *main.mtl* also produces an xta file containing code in UPPAAL syntax 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 verification tool as template in order to check the deadlock and reachability of our model using queries in a specific syntax followed by UPPAAL. **Java Services:** Java Services are developed using three main classes, *JavaServices*, *ExcelWrite* and *Model*. *Model* class only used to get the input IFML model and instantiates the IFML model. *JavaServices* is the main class we have used to store every detail needed for our transition matrices as Accelo does not provide facility of storing things in easy way inside the mtl file. The functions defined in *JavaServices* class can then be used inside *main.mtl* file using queries. *ExcelWrite* is only used to shape our matrices in the form of excel sheets. One of the matrices contain 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 us 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. 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.

Chapter 5

Validation

CHAPTER 5: VALIDATION

In this chapter, the applicability and validity of our proposed approach is presented with the help of two detailed case studies. Details and validation of Online Auctions case study is given in **Section 5.1** and Library case study is presented in **Section 5.2**.

5.1. Online Auctions Case Study

Online Auctions case study has been explained and validated using four sections. **Section 5.1.1** covers the requirement specification for a website named as Online Auctions. **Section 5.1.2** presents the UML domain modeling and IFML modeling of this case study in Eclipse editor using its IFML plugin. Lastly, the transformation and verification of the case study modeled in IFML has been provided in **Section 5.1.3** and **Section 5.1.4** respectively.

5.1.1. Requirement Specification

Following are the details of the web pages and their specifications included in the Online Auctions case study. Generic specifications are given for example a small separate portion inside a web page is known as division or div but the user may refer to it as section. So, modeling needs to be done intelligently.

MASTER PAGE: The Online Auctions website should have a landmark page called Master page which should be accessible from all other pages that exist. 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 which will redirect the page to itself. 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.

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.

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.

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.

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.

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.

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.

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.

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

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.

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 sections. Category overview page should be accessible from Home page.

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.

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.

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.

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 two 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.

5.1.2. Modeling

Our tool provides the option to open IFML Editor. By clicking on it, Eclipse environment is opened and IFML model can be designed by creating a new project for IFML modeling. Now in eclipse, 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. Domain model of the Online Auctions case study is shown in **Figure 13**.

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.

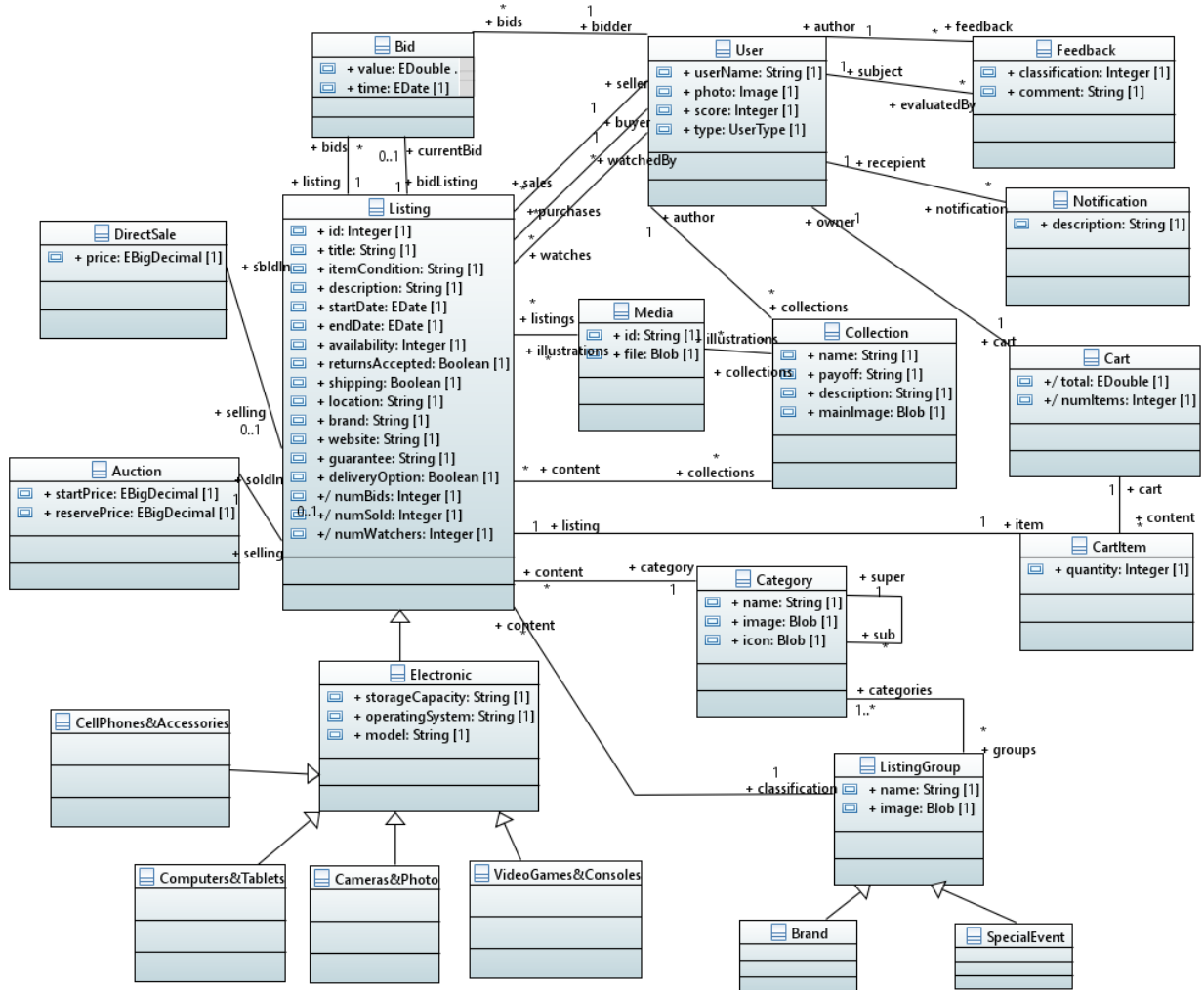


Figure 13: Online Auctions Domain Model

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. Specifications defined in **Section 5.1.1** 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. Due to complexity, parts of IFML diagram of Online Auctions case study is shown in several subsequent figures.

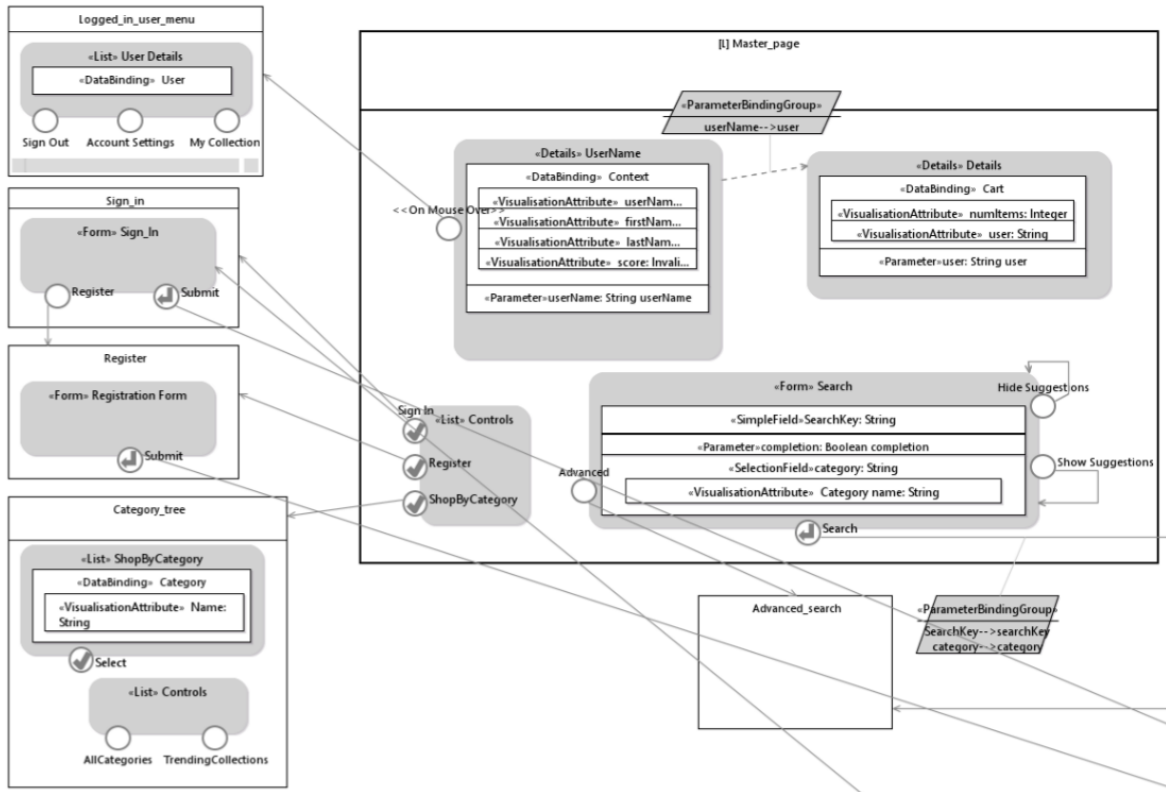


Figure 14: Online Auctions IFML Model (Diagram 1 of 5)

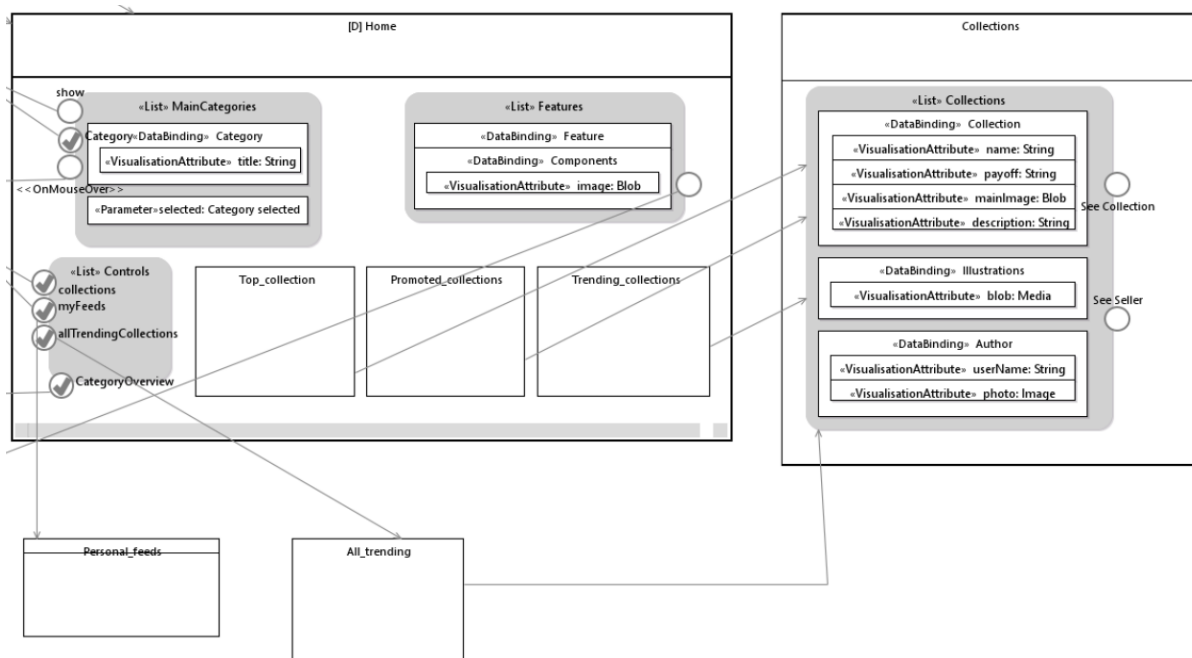


Figure 15: Online Auctions IFML Model (Diagram 2 of 5)

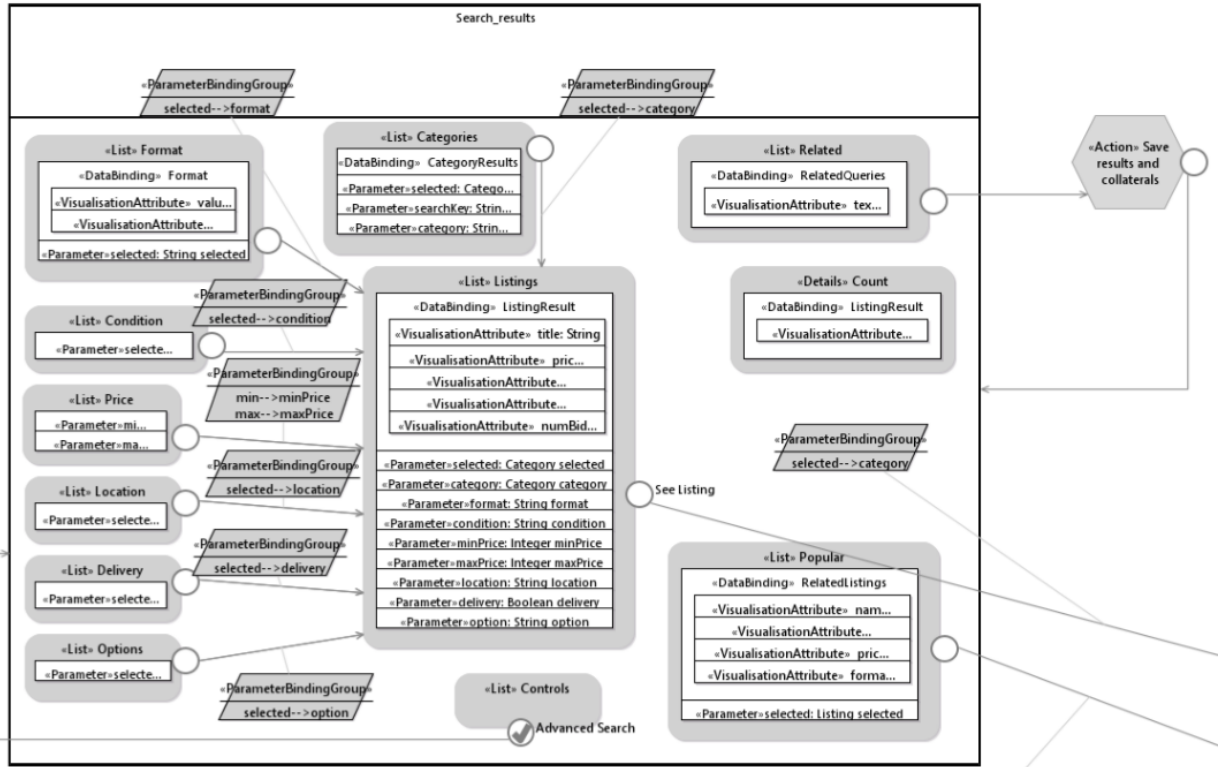


Figure 16: Online Auctions IFML Model (Diagram 3 of 5)

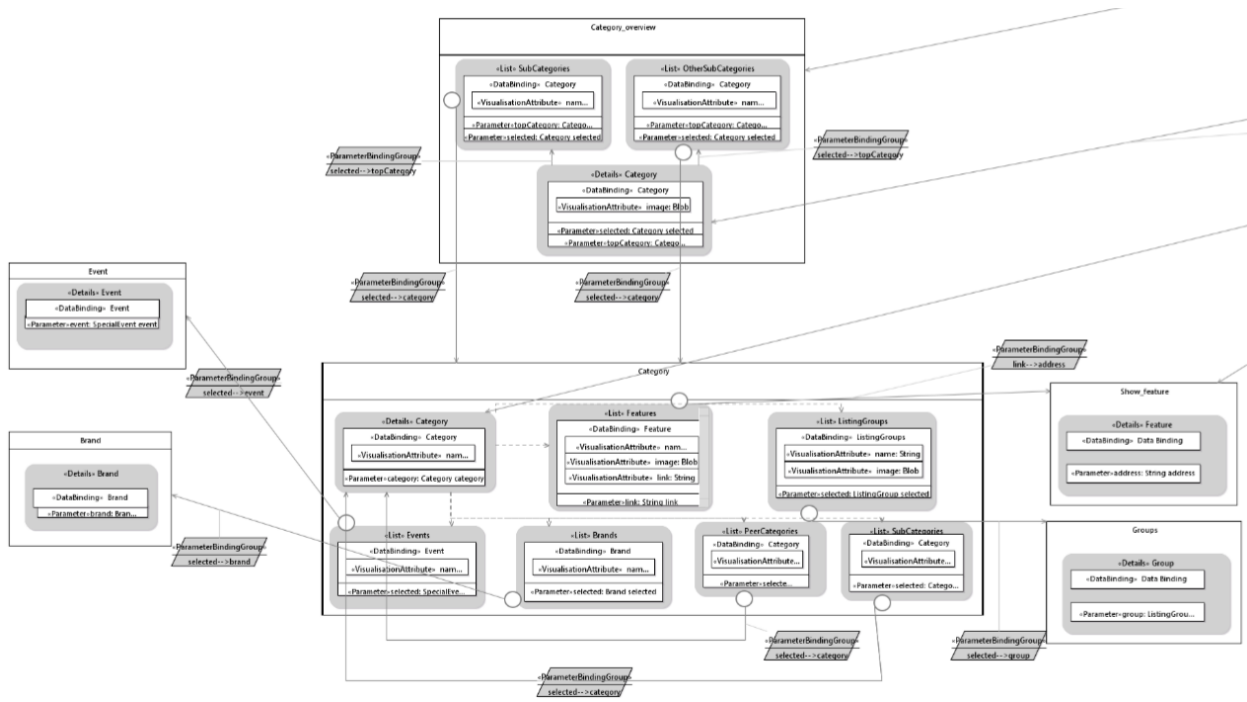


Figure 17: Online Auctions IFML Model (Diagram 4 of 5)

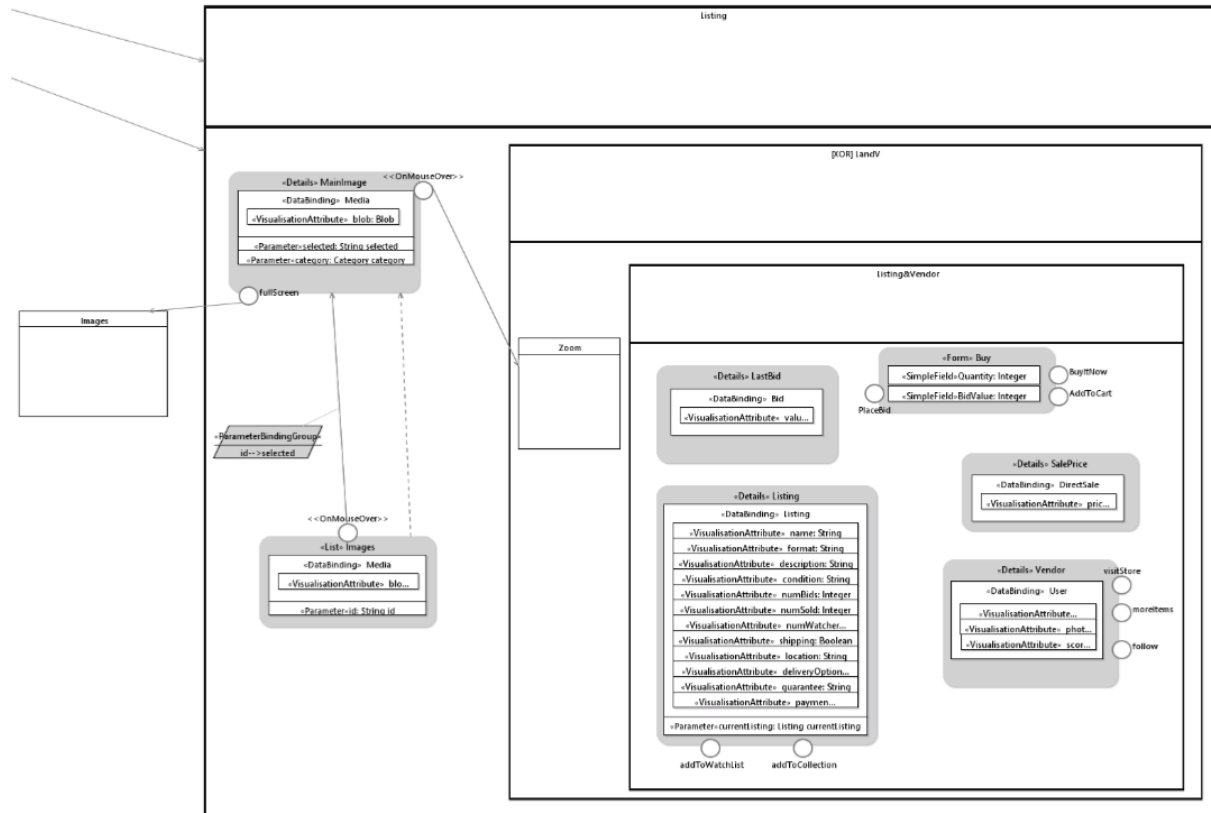


Figure 18: Online Auctions IFML Model (Diagram 5 of 5)

5.1.3. Navigation Model and Test Case Generation

In Figure 19, 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 doesn't 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.

1. 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 Figure 20)
2. 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 Figure 21)

3. Test case document (.txt) with detailed UI test cases related to Forms, Lists and Details is generated. (One test case from the Test case document is shown in **Figure 22**)
4. A text file (.xta) is generated as last output. This file contains UPPAAL code of navigation model. This file acts as a template and can be imported in UPPAAL tool for verification of deadlock and reachability.

Status displays that applied transformation was successfully performed.

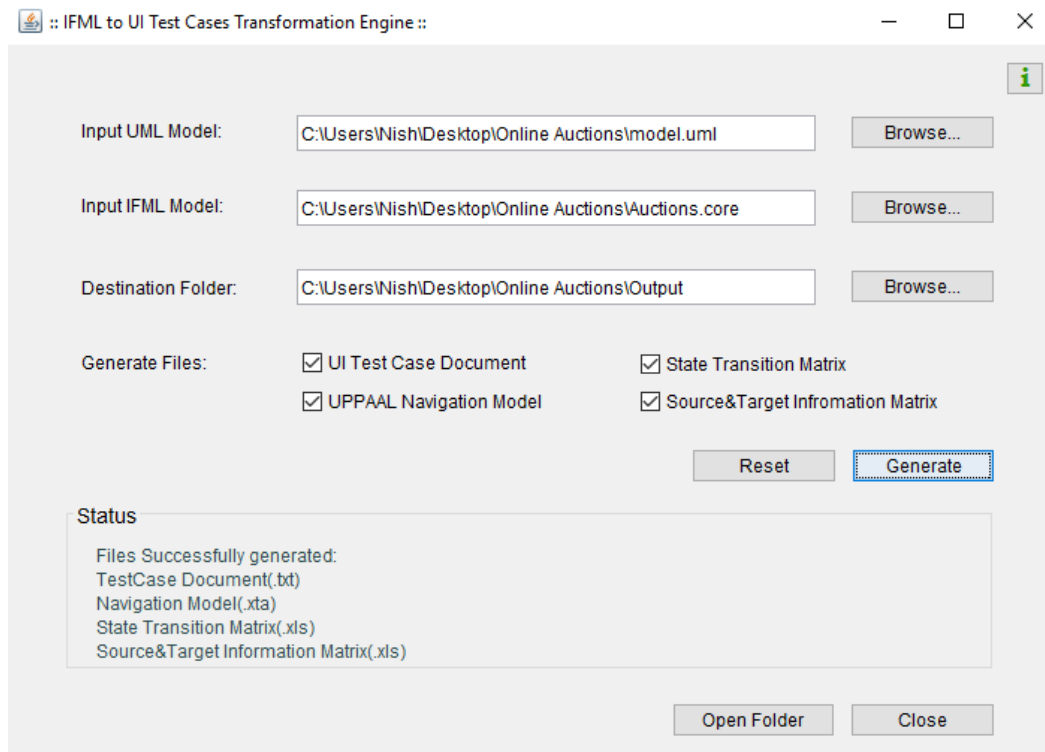


Figure 19: Transformation for Online Auctions Model

	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_i	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 20: 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 21: Source&Target Information Matrix

```

-----
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 22: Test Case for Search Form

5.1.4. Automated Navigation Verification

UPPAAL model checker is the tools selected for verification purpose. When we load the xta template file in UPPAAL (**Figure 23**), 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:

- **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

If the property is satisfied, it means that the model is deadlock free. **Figure 24** shows that our output model satisfies both these properties. Hence, our model is reachable and deadlock free.

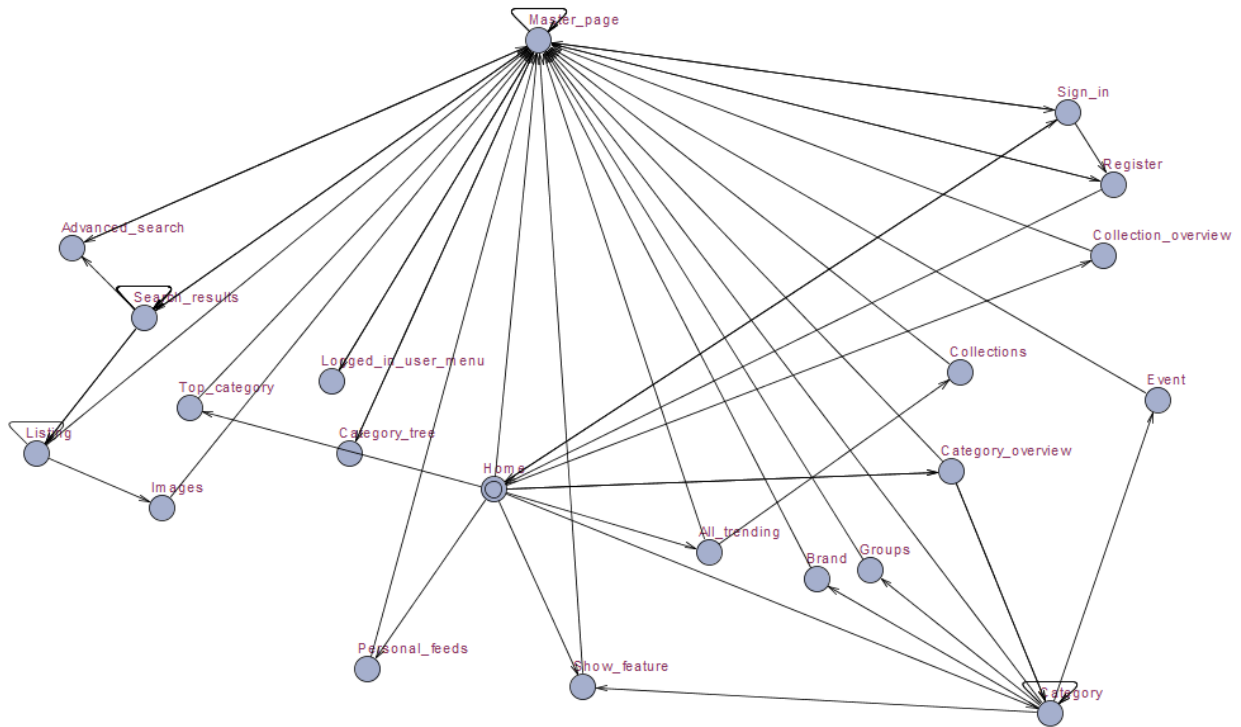


Figure 23: UPPAAL Model for Online Auctions

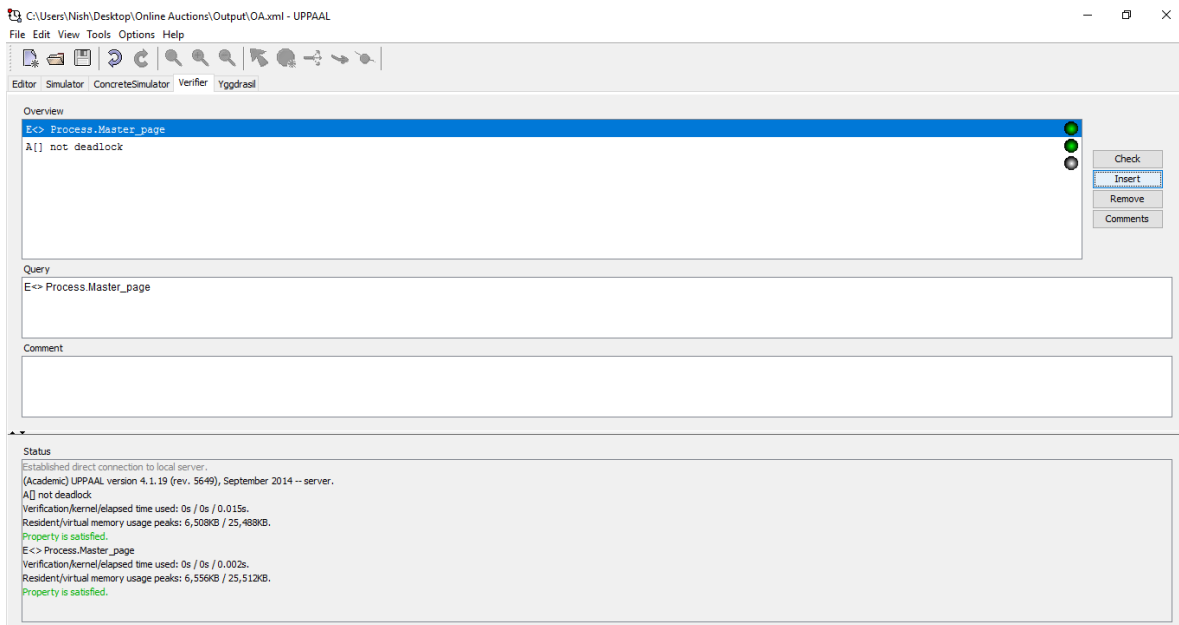


Figure 24: Deadlock and Reachability verification

5.2. Library Case Study

A chunk of Library case study has been used as second case study for validation. **Section 5.2.1** covers the requirement specification for a part of library web application. **Section 5.2.2** presents the UML domain modeling and IFML modeling of this case study in Eclipse editor using the IFML plugin. Lastly, the transformation and verification of the case study modeled in IFML has been provided in **Section 5.2.3** and **Section 5.2.4** respectively.

5.2.1. Requirement Specification

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

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.

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.

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.

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.

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

5.2.2. Modeling

As we have taken a simple feature of the case study, only one class is needed in domain model. Book class contains important data about book i.e. title, author, year and description. These attributes are later used in the IFML model. Book class shown in **Figure 25** has been designed using Papyrus editor.

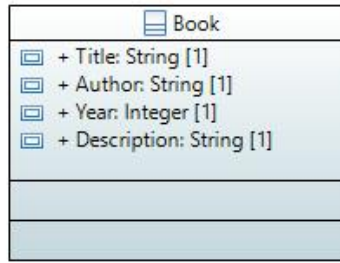


Figure 25: Library Domain Model

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 in **Section 5.2.1**. The pages mentioned in specifications are modeled as view containers. Lists, details and forms are modeled as extensions of view components. **Figure 26** shows the IFML model for the part selected from library case study.

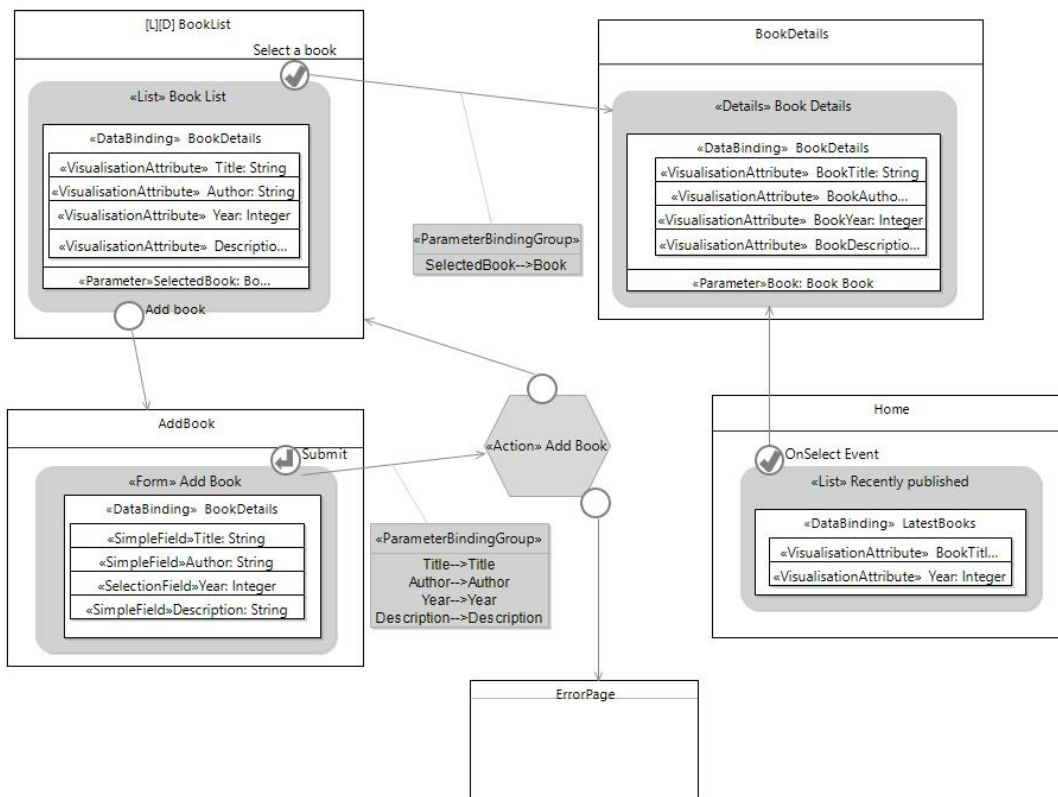


Figure 26: Library IFML Model

5.2.3. Navigation Model and Test Case Generation

Library UML (.uml) and IFML (.core) models are given as input (**Figure 27**). In destination folder, we have specified Desktop. When we click on “Generate” button, the input IFML model is transformed into State Transition Matrix (**Figure 28**), Source&Target Information Matrix (**Figure 29**), a test case document (**Figure 30**) and a text file (.xta) containing UPPAAL code. On successful transformation “Status” displays that the outputs have been generated as shown in **Figure 27**.

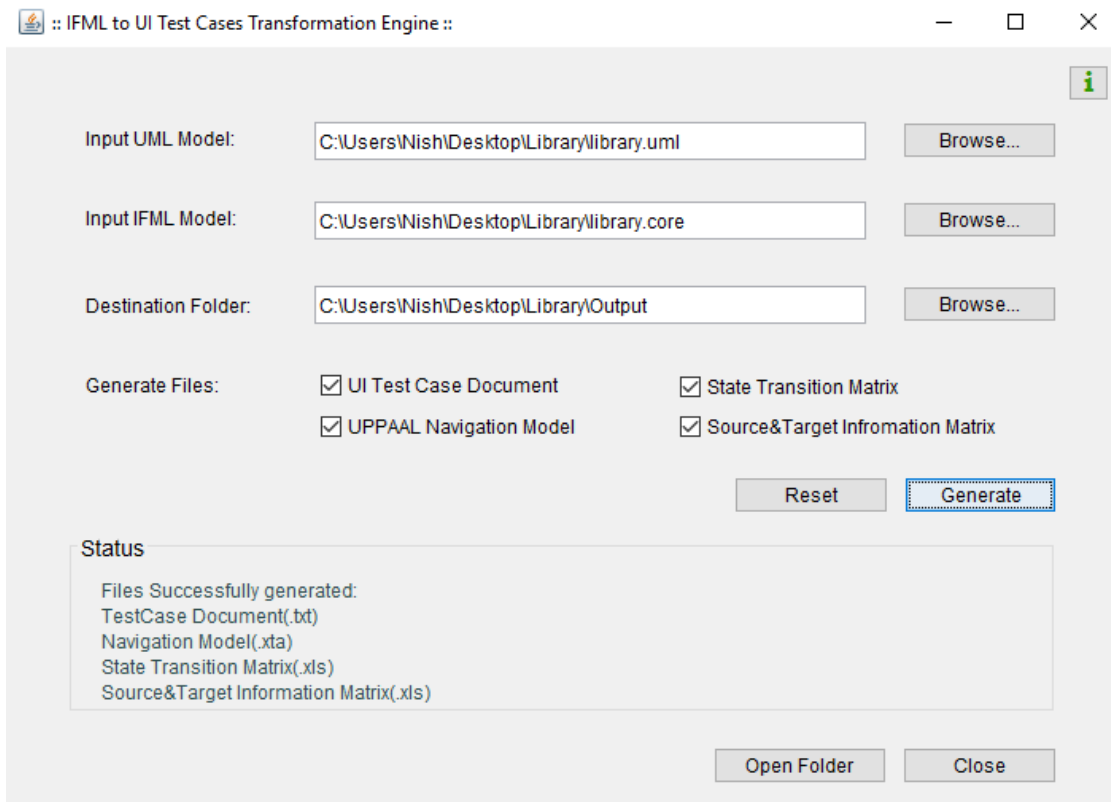


Figure 27: Transformation for Library Model

	A	B	C	D	E	F
1		BookDetails	BookList	AddBook	Home	ErrorPage
2	BookDetails	F	F	F	F	F
3	BookList	T	T	T	T	T
4	AddBook	F	F	F	F	F
5	Home	T	F	F	F	F
6	ErrorPage	F	F	F	F	F

Figure 28: Library State Transition Matrix

	A	B	C	D	E	F
1	Source Page	LandMark	Default	XOR	Parameter	Target Page
2	BookDetails	false	false	false	null	BookList
3	BookList	true	true	false	null	BookList
4	AddBook	false	false	false	null	BookList
5	Home	false	false	false	null	BookList
6	ErrorPage	false	false	false	null	BookList
7	BookList	true	true	false	Book	BookDetails
8	BookList	true	true	false	null	AddBook
9	Home	false	false	false	null	BookDetails

Figure 29: Library Source&Target Information Matrix

```

-----
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 30: Test Case for Book Details

5.2.4. Automated Navigation Verification

When the xta template file is loaded in UPPAAL, it shows us the navigation model for library system (**Figure 31**). Reachability and deadlock properties are verified for library system using UPPAAL property checking syntax.

E<> Process.BookList

A[] not deadlock

Figure 32 shows that our output model satisfies both these properties. Hence, our library model is reachable and deadlock free.

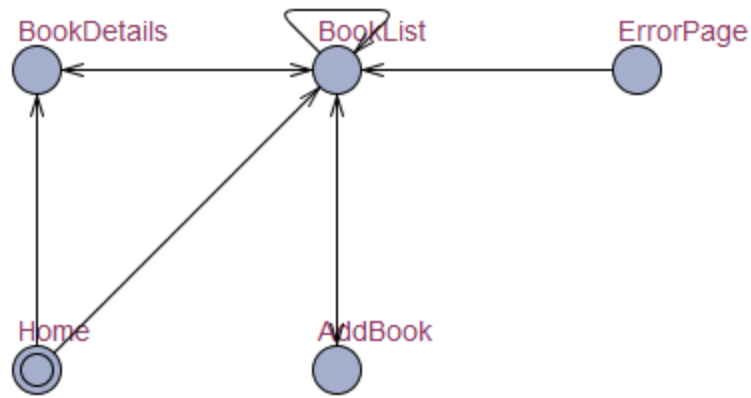


Figure 31: UPPAAL Model for Library System

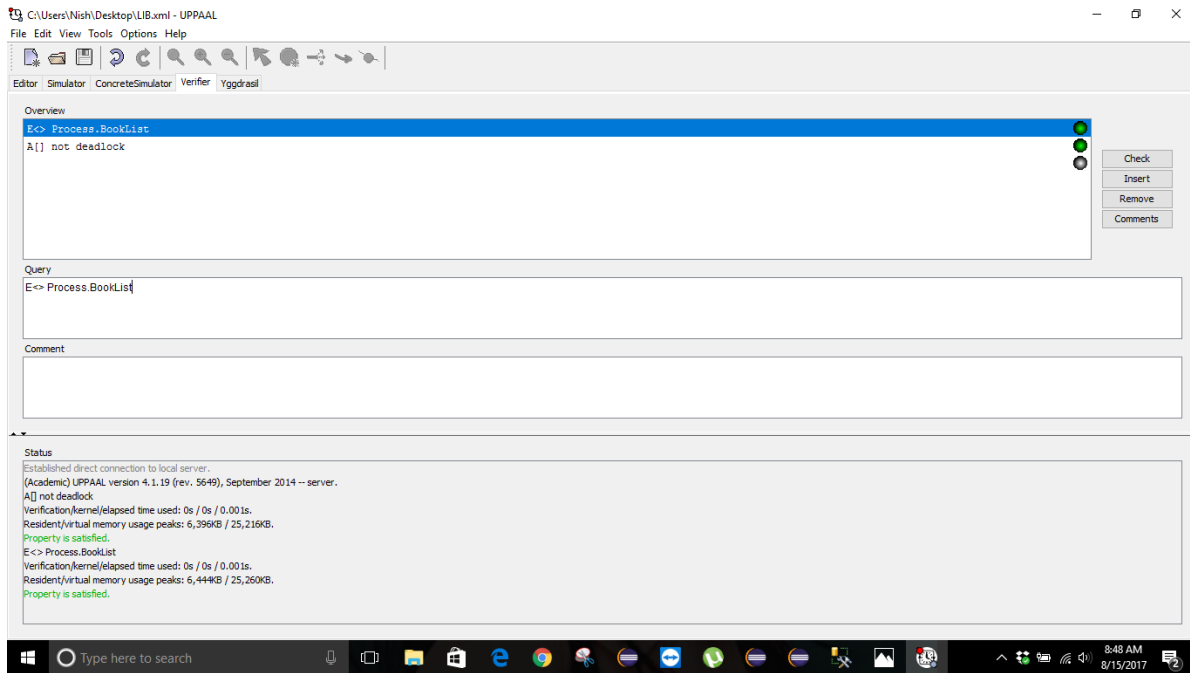


Figure 32: Deadlock and Reachability verification for Library System

Chapter 6

Discussion and Limitation

CHAPTER 6: DISCUSSION AND LIMITATION

This section presents a detailed discussion on this research work (**Section 6.1**) and limitations to the research are also presented in **Section 6.2**.

6.1. Discussion

From this research, it has been analyzed that there is a very limited amount of research work done on IFML in the area of model based UI testing and the available research work did 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 the real testing artifacts that a developer will need in order to develop the right product meeting the user specified UI requirements and a tester will encounter while dealing with testing of actual web application. 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 while navigation which is detailed in the Source and Target matrix produced by the 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.

IFML has been inspired from WebML but is not fully evolved yet. Only two tools “WebRatio” and “Eclipse IFML plugin” are available in market which provide IFML modeling. Even these tools are not mature enough and there were still some problems encountered during IFML modeling so we can say that IFML has not been fully covered in our research at the present time. One of the many issues is the issue of nested containers. Both the tools did not

allow nesting for more than 3 levels. Even though the events are allowed on view containers, but both these tools did not allow them. Navigation flows are not allowed inside nesting hierarchy by the tools. There are few other limitations to IFML modeling found using these tools but as the language is evolving and a lot of work is being done on its tool support, we are hoping that these issues will be resolved soon.

Two case studies of different sizes have been selected in order to validate our proposed approach. First case study we selected was a very detailed case study on 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. The purpose of taking into account these two case studies was to validate the testing mechanism for web applications of different sizes.

6.2. Limitations

As we have taken the first step to automation testing for IFML, there are a few limitations to our work. IFML has a lot of potential but due to limited amount of time and resources, we have currently only selected limited core metamodel elements i.e. *ViewContainer*, *ViewComponent*, *NavigationFlow* etc. and a few important extensions metamodel elements i.e. *Form*, *List* and *Details* etc. There are many IFML constructs that we have not yet considered for example *Menu*, *Window*, *Action*, *Module* and *ActivationExpression* etc. on which we intend to work in future.

Chapter 7

Conclusion and Future Work

CHAPTER 7: CONCLUSION AND FUTURE WORK

The proposed approach gives a big overhead to UI automation testing using IFML models to capture the content, structure and control flow of the web application front-end. Generation of UI test cases in the early stages of development cycle will allow developers and testers to develop the right product by embedding the quality in it from the beginning of development and by making the testing process less complex. The proposed approach makes use of MDE for UI Testing through Aceleo transformations based on different rules resulting in test cases which cover the navigational aspects along-with the structural aspects of web application user interface. The approach has been validated and verified using two case studies varying in sizes from simple to compound. The results of our proposed approach proved the potential and viability of IFML for UI test case generation.

Future work includes improving and extending this approach in order to support other important IFML constructs like *Module*, *Action*, *Menu*, *Context* and *Expression* etc. The approach can be expanded by integration of UML behavior diagrams with IFML in order to provide fully functional test cases for business processes along-with the UI testing.

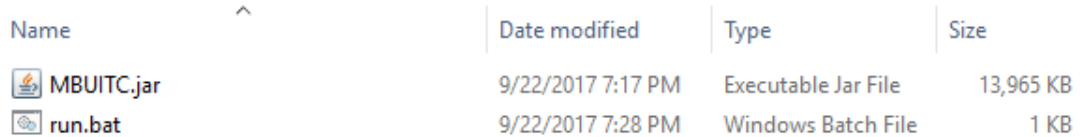
APPENDIX A

USER MANUAL

1. Download Instructions

1.1. Model Based UI Test Case Generator (MBUITC)

Extract **MBUITC Generator.zip** file. You will find “MBUITC Generator” Folder. In the “MBUITC Generator” folder, you will find two files shown in **Figure 1** below.





Name	Date modified	Type	Size
 MBUITC.jar	9/22/2017 7:17 PM	Executable Jar File	13,965 KB
 run.bat	9/22/2017 7:28 PM	Windows Batch File	1 KB

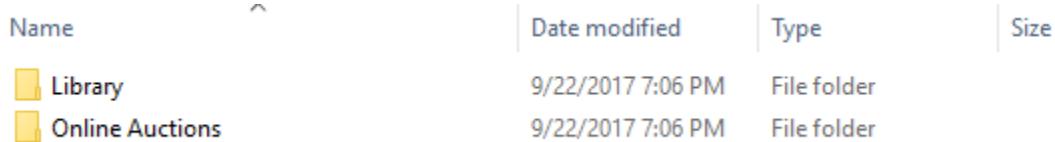
Figure 1: Files in “MBUITC Generator” folder

Click “run.bat” file to execute MBUITC Generator.

1.2. Sample Case Studies

Extract **Sample-CaseStudies.zip** file. You will find “Online-Auctions” and “Library” folders as shown in **Figure 2**.

Each folder contains UML and IFML models for the respective case study developed in Eclipse using Papyrus and IFML plugin.





Name	Date modified	Type	Size
 Library	9/22/2017 7:06 PM	File folder	
 Online Auctions	9/22/2017 7:06 PM	File folder	

Figure 2: Sample Case Study folder

You can use the existing UML and IFML models to generate complete UI test cases or you can update the IFML model to include modeling of more web pages.

2. Prerequisites for MBUITC Generator

It is mandatory to install **Java Runtime Environment (JRE) version 8 or above** in order to execute MBUITC Generator through run.bat file.

We have tested MBUITC Generator on Windows 8, Windows 8.1 and Windows 10. However, we are confident that MBUITC Generator can also be executed on previous versions of Windows.

3. Execution of MBUITC Generator

Click “**run.bat**” file in order to execute the MBUITC Generator. (See section for complete details).

The main interface of MBUITC Generator is shown in **Figure 3**.

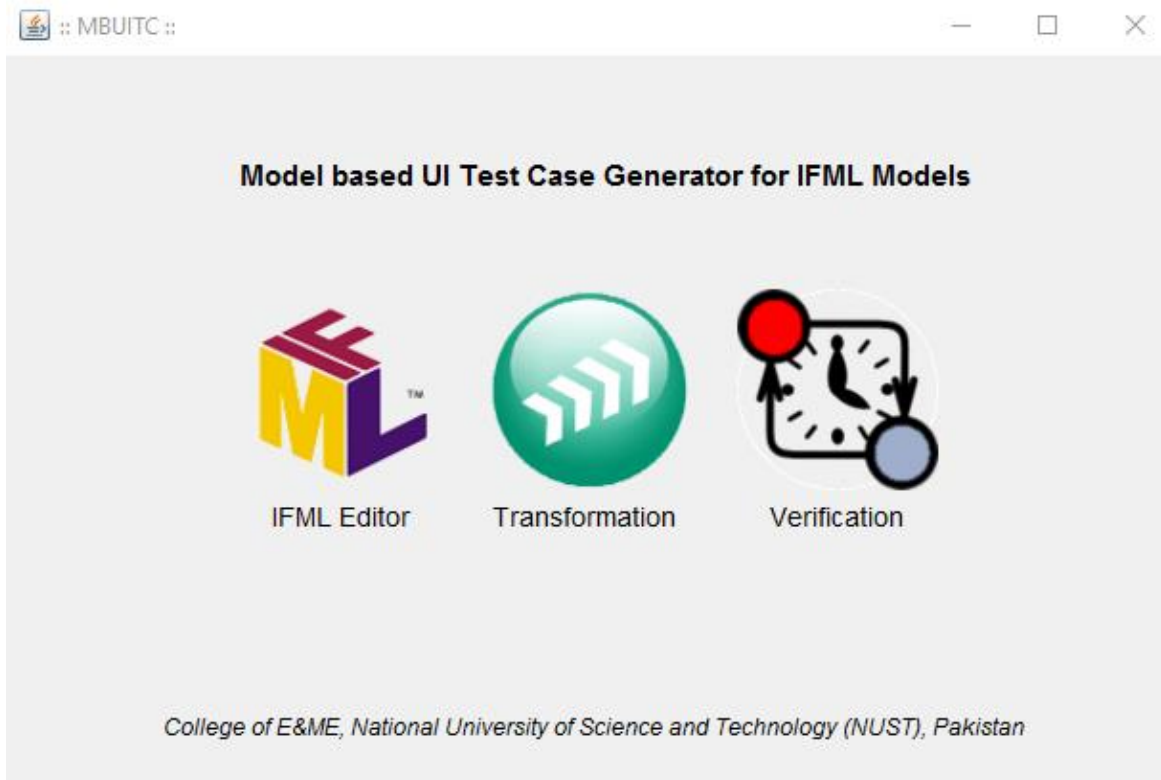


Figure 3: Main interface of MBUITC Generator

The MBUITC contains three main functionalities.

- **IFML Editor:** The IFML Editor allows you to model your own web application using Interaction Flow Modeling Language (IFML).
- **Transformation:** Transformation engine contains the main functionality of generating UI test cases from UML and IFML models. Details of the execution of transformation engine are given in next section.
- **Verification:** Automated navigation verification of the navigation model generated by the transformation engine is done using UPPAAL model checker.

3.1. IFML Editor

By clicking on “**IFML Editor**” Eclipse environment is opened and in order to allow IFML modeling “Eclipse IFML plugin” must be installed.

3.2. Transformation

By clicking on “**Transformation**” in the main interface, interface for “**IFML to UI Test Cases Transformation Engine**” is opened (shown in **Figure 4**).

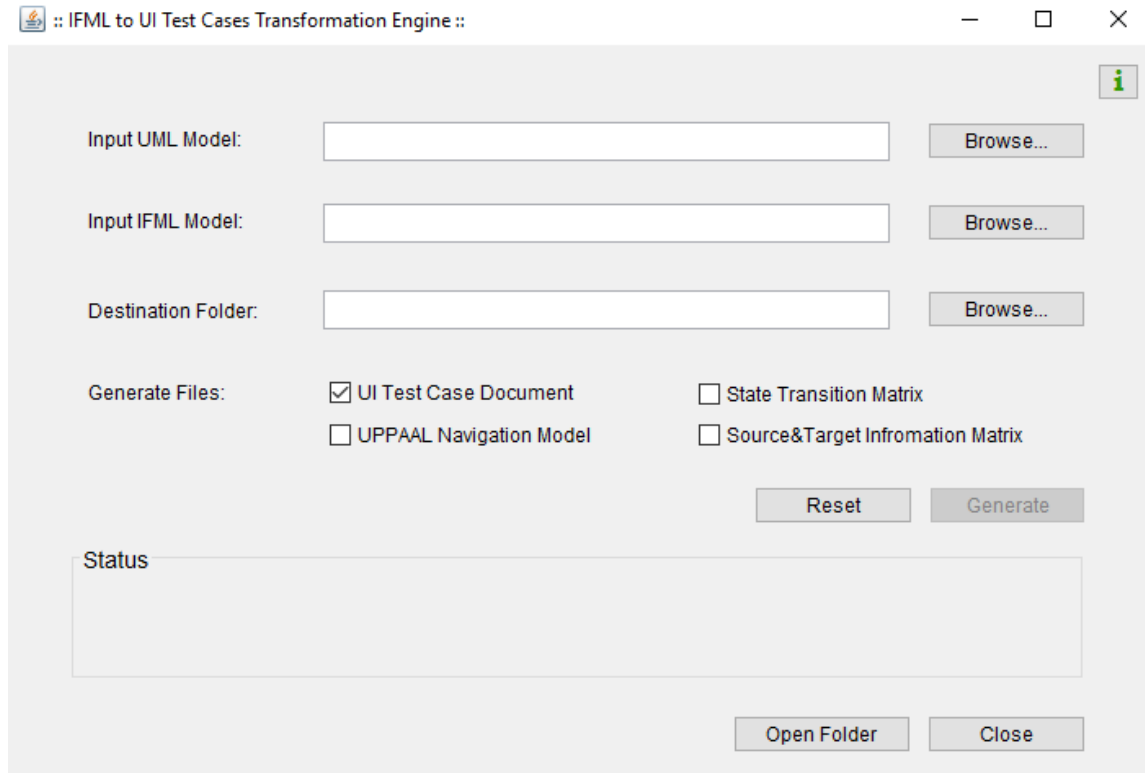


Figure 4: Interface for IFML to Test Cases Transformation Engine

Input UML Model: Browse button is used to select the UML model for the case study.

Input IFML Model: Browse button is used to select IFML model for which one desires to perform UI Testing.

Destination Folder: Browse button is used to specify the destination folder for the generated files.

Generate Files: User can select the required files from the given four options by checking the checkbox.

Reset: This button clears all the current selections to define new configurations.

Generate: This button transforms the selected UML and IFML models into the required testing artifacts. It is mandatory to fill all the above fields in order to click generate button.

Status: This displays the status of current transformations i.e. List of generated files or Files Generated with Errors (in case of any problem in transformation).

Open Folder: This button is used to open the folder where output folder containing the generated files is placed.

Close: This button closes the interface.

The UML and IFML models can be selected using browse button against each selection. **Figure 5** shows the selection of Library model using browse button.

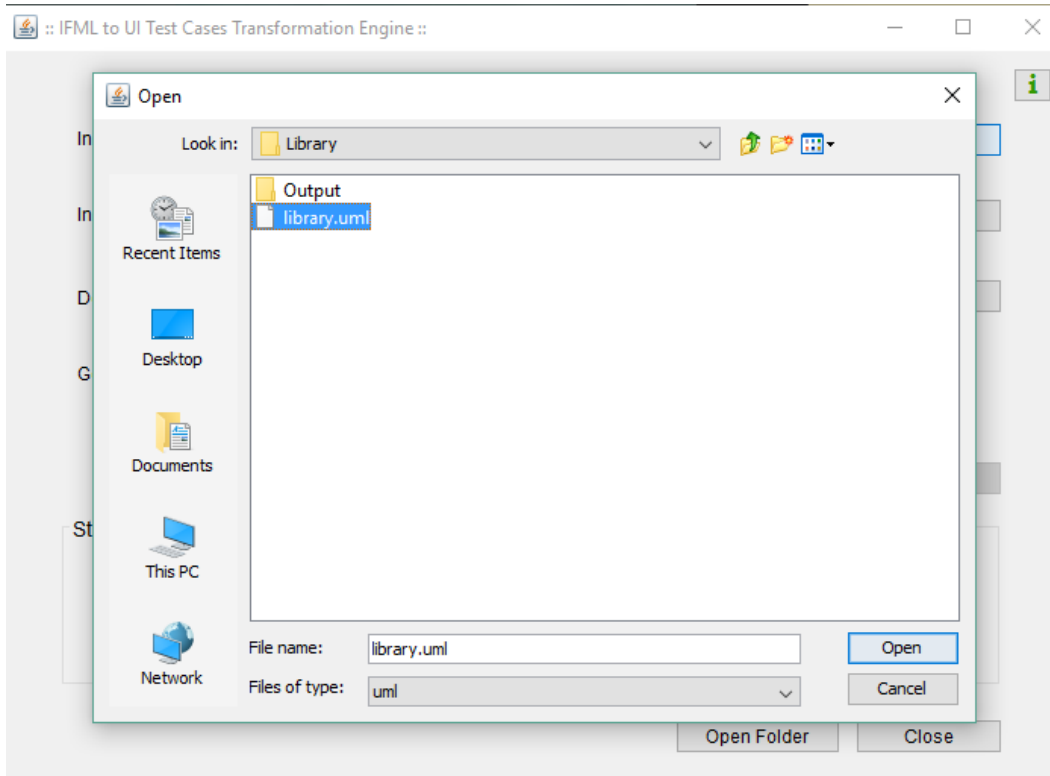


Figure 5: Selection of Library UML model using browse button

The Library UML and IFML models can be transformed into testing artifacts through *Generate* Button as shown in **Figure 6**.

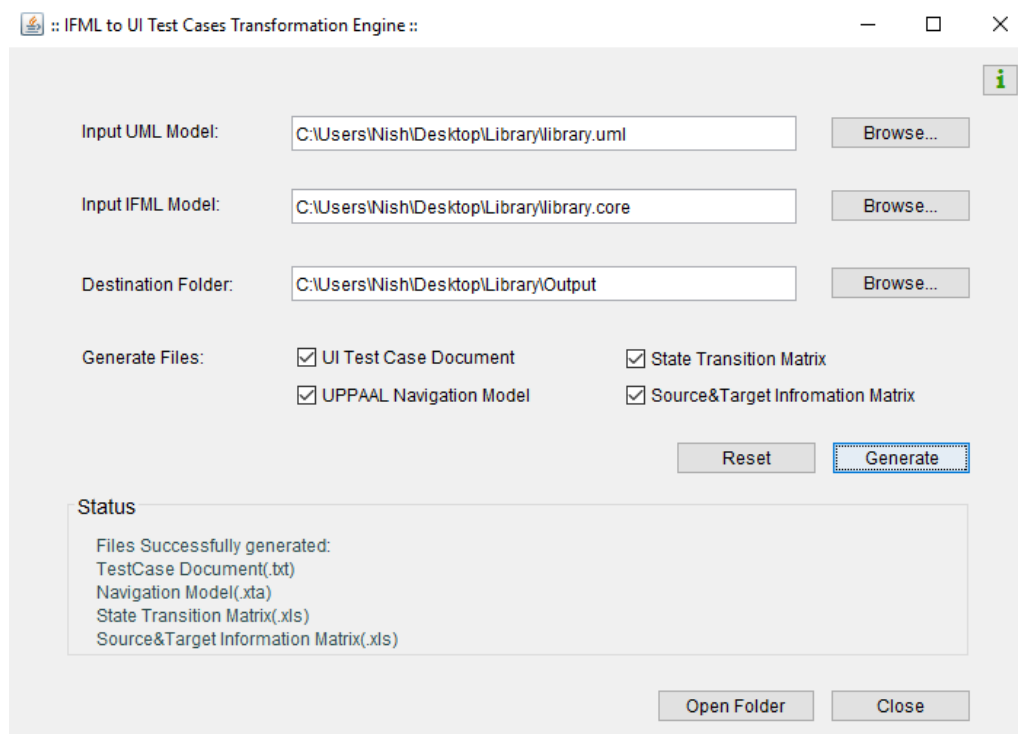


Figure 6: Generating UI Testing Artifacts for Library Model

The screenshot for the output folder containing generated files is shown in **Figure 7**.

Name	Date modified	Type	Size
Source&TargetInformationMatrix.xls	9/19/2017 9:51 PM	Microsoft Excel 97...	14 KB
StateTransitionMatrix.xls	9/19/2017 9:51 PM	Microsoft Excel 97...	14 KB
Template.xta	9/19/2017 10:06 PM	XTA File	1 KB
TestCase.doc.txt	9/19/2017 9:51 PM	Text Document	4 KB

Figure 7: Output folder containing generated files

3.3.Verification

One of the generated files “**Template.xta**” is used for navigation verification of the given IFML model. From the main interface of MBUITC Generator, click on “**Verification**”, It will open UPPAAL model checker tool. The xta file is imported in the tool through File>Import Template. Screenshot for selection of Template.xta from the output folder for Library case study in UPPAAL is given in **Figure 8**.

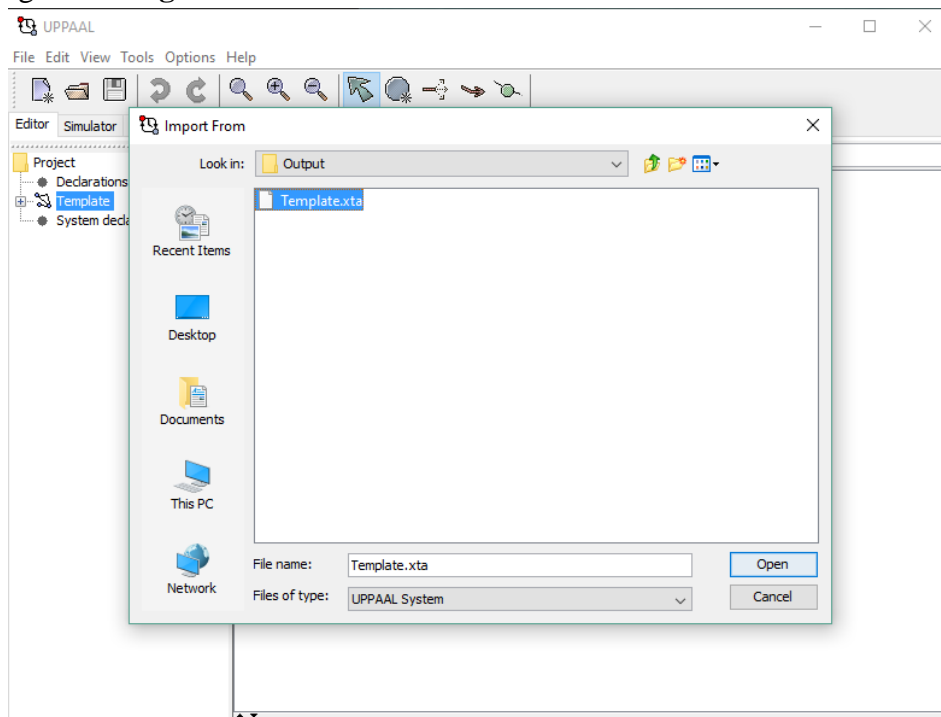


Figure 8: Importing Template.xta for Library case study

The imported template is opened as Timed Automata in UPPAAL where we can verify the navigation properties i.e. reachability and deadlock freedom for our model. **Figure 9** shows the template opened in UPPAAL model checker.

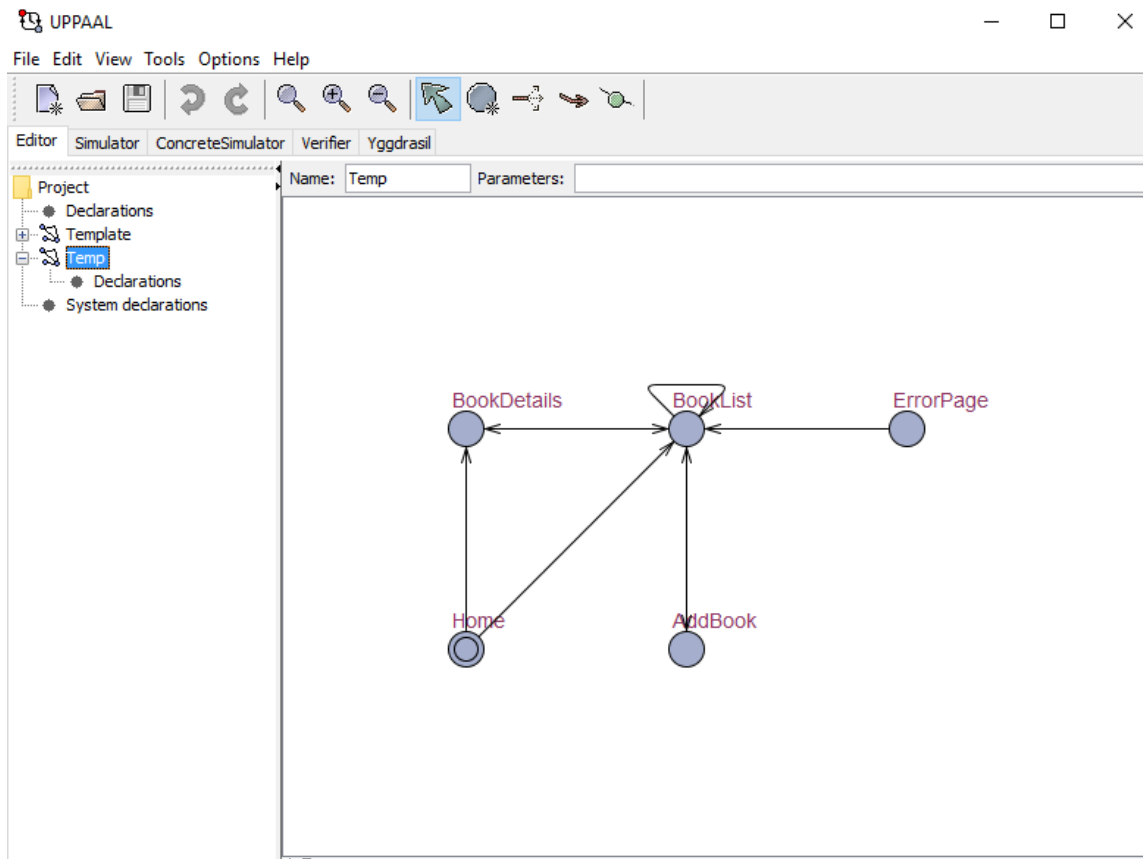


Figure 9: Template.xta forLibrary case study opened in UPPAAL

REFERENCES

- [1] Raluca, M., Cristina, C. S., Helene, L. G., Paul, P., 2015,“A Research Overview of Tool-Supported Model-based Testing of Requirements-based Designs”, Journal of Advances in Computers
- [2] Hamza, S.,Hélène, L. G., 2014,“Ralf Bogusch; Mathieu Acher; Benoit Baudry: Deriving Usage Model Variants for Model-Based Testing: An Industrial Case Study”, 19th International Conference on Engineering of Complex Computer Systems (ICECCS)
- [3] Justyna, Z., Ina, S., Pieter, J. M., 2011,“Model-Based Testing for Embedded Systems”, CRC Press
- [4] Jacky, J., 2011,“PyModel: model-based testing in python”, Proceedings of the Python for Scientific Computing Conference
- [5] Dimitris, D., Konstantinos, B., Florentin, I., 2012,“JSXM: a tool for automated test generation”, Software Engineering and Formal Methods, pp 352–366
- [6] Tommi, T., Mika, K., Julian, H., 2011,“Experiences of system-level model-based GUI testing of an Android application”, IEEE Fourth International Conference on Software Testing, Verification and Validation (ICST 2011), pp 377–386
- [7] Paulo, S., 2016, “Practical Programming, Validation and Verification with Finite-State Machines: a Library and its Industrial Application”, IEEE/ACM 38th IEEE International Conference on Software Engineering Companion
- [8] Yu, Z., Jidong, G., Pengcheng, Z., and Weigang, W., 2016,“Model based verification of dynamically evolvable service oriented systems”, Science China Information Sciences
- [9] Monalisa, S., P.V.R. Murthy, Sylvia, J., Andreas, U., 2010,“Model-based testing in industry: a case study with two MBT tools”, Proceedings of the 5th Workshop on Automation of Software Test, pp 87–90.
- [10] Dehla, S., Stephan, W., 2010,“ParTeG-integrating model-based testing and model transformations”, Software Engineering, pp 23–24
- [11] Roberto, S. S. F.,Christof, J. B., 2012, “An integrated model-driven approach for mechatronic systems testing”, IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST 2012), pp 447–456

- [12] Zhenying, J.,Xiao, W.,Zeqian, D., 2017,“Ming Mu: Optimal Test Case Generation for Simulink Models Using Slicing, Software Quality”, IEEE International Conference on Reliability and Security Companion (QRS-C)
- [13] Barbara, K., 2004,“Procedures for Performing Systematic Reviews”, Keele University TR/SE-0401/NICTA, Technical Report 0400011T
- [14] Guo, L., Hua, Q.Y., Wang, P., Yu, K., Wang, K., Xu, J., 2015,“A Model-Based Responsive Web User Interface Development Method”, IEEE Proc. ICSESS
- [15] Piero, F., Sara, C., Alessandro, B., Giovanni, T.C.,2010, “Engineering Rich Internet Applications with a Model-Driven Approach”, ACM Transactions on the Web (TWEB), Vol. 4(2)
- [16] Marco, B., Andrea, M., Eric, U., 2014,“Extending the Interaction Flow Modeling Language (IFML) for Model Driven Development of Mobile Applications Front End”, Springer Proc. MobiWIS. Lecture Notes in Computer Science,pp 176-191
- [17] Eric, U., Marco, B., 2016,“Model Driven Development Approaches for Mobile Applications: A Survey”, Springer Proc. MobiWIS. Lecture Notes in Computer Science, pp 93-107
- [18] Thiago, G., Rosana, T.V.B., 2016,“Model-Oriented Web Services. IEEE Symposium on SOSE
- [19] Adrian, F., Silvia, A., Emilio, I., Maristella, M., 2012,“Further analysis on the validation of a usability inspection method for model-driven web development”, ACM-IEEE International Symposium on ESEM
- [20] Jose, L.H.A., 2015,“Model-Driven Web Applications. IEEE Proc. SAI
- [21] Mardiana, Keijiro, A., Yoichi, O., 2011,“MDA and SOA Approach to Development of Web Application Interface”, IEEE Proc. TENCON
- [22] Mohammed, A.O.M., Mohd, F.B.H., Jafreezal, B.J., Lukman, A.R., 2013,“Enhanced Approach for Developing Web Applications Using Model Driven Architecture”,IEEE Proc. ICRIIS
- [23] Mohammed, A.O.M., Mohd, F.B.H., Jafreezal, B.J., Lukman, A.R., 2014,“WSDMDA: An Enhanced Model Driven Web Engineering Methodology”, IEEE Proc. ICCSCE

- [24] Sarra, R., Mohammad, E., and Samir, M., 2015, "A Model Driven Approach to generate Graphical User Interfaces for Rich Internet Applications Using Interaction Flow Modeling Language", IEEE Proc. ISDA, Marrakech, Morocco
- [25] Pedro, V., Vicente, P., 2011, "A Survey of Requirements Specification in Model-Driven Development of Web Applications", ACM Transactions on the Web (TWEB), Vol. 5(2)
- [26] Fabio, P., Christian, S., 2011, "Model-Based Customizable Adaptation of Web Applications for Vocal Browsing", ACM Proc. Of SIGDOC
- [27] Marco, B., Andrea, M., Mirco, F., Henry, M., 2016, "A Model-Based Method for Seamless Web and Mobile Experience", ACM Proc. of the 1st International Workshop on Mobile Development
- [28] Carlo, B., Marco, B., Thanas, K., Andrea, M., Eric, U., 2017, "Integrating Modeling Languages and Web Logs for Enhanced User Behavior Analytics", ACM Proc. WWW '17 Companion
- [29] Roberto, A., Aldo, B., Stefano, B., Marco, B., 2015, "Model-driven development of cross-platform mobile applications with WebRatio and IFML", Proc. MOBILESoft '15
- [30] Jose, M.R., Julian, G., Gustavo, R., Esteban, R.L., Francisco, M., Martin, G., 2014, "Mockup-Driven Development: Providing agile support for Model-Driven Web Engineering", Elsevier Journal on Information and Software Technology, Vol. 56(6)
- [31] Fransisco, J.D.M., Maria, J.E., Manuel, M., M, Ross., Geoff, S., 2012, "Quality evaluation for Model-Driven Web Engineering methodologies", Journal of Information and Software Technology, Vol. 54(11)
- [32] Marco, B., Piero, F., 2014, "Large-scale Model-Driven Engineering of web user interaction: The WebML and WebRatio experience", Journal of Science of Computer Programming, Vol. 89(B)
- [33] Jose, L.H.A., Pablo, C.D.B., 2013, "A model-driven approach to develop high performance web applications", Journal of Systems and Software. Vol. 86(12)
- [34] Fabio, P.B., Raquel, M.P., Toacy, C.O., 2016, "Automated design of multi-layered web information systems", Journal of Systems and Software, Vol. 117
- [35] Jose, I.P., Natalia, J., Francisco, V., Óscar, P., 2015, "A framework to identify primitives that represent usability within Model-Driven Development methods", Journal on Information and Software Technology, Vol. 58

- [36] Muhammad, U., Muhammad, Z.I., Muhammad, U.K., 2017,“A product-line model-driven engineering approach for generating feature-based mobile applications”,Elsevier Journal of Systems and Software, Vol. 123
- [37] Humberto, L.A., Elias, A.N.S., Renata, P.M.F., 2015,“A model-driven development for creating accessible web menus”, Proc. Computer Science DSAI
- [38] Dhiraj, G., Madhu, N., 2015,“Review: Analysis of Aspect Orientation and Model Driven Engineering for Code Generation”, Proc. Computer Science ICACTA
- [39] Yassine, R., Youssef, H., Abdelaziz, M., 2016,“Model Transformation with ATL into MDA from CIM to PIM Structured through MVC”, Proc. Computer Science SEIT
- [40] Vikas, S., Rajesh, B., 2014,“Model based Test Cases Generation for Web Applications”, Springer IJCA, Vol. 92(3)
- [41] Zef, H., Lennart, C.L.K., Danny, M.G., Eelco, V.,2010, Code generation by model transformation: a case study in transformation modularity”, Springer Journal of Software & System Modeling, Vol. 9(3)
- [42] Vanessa, N., Veronica, C., Fernando, L., Roberto, F., Claudio, G., 2016,“Model Driven Architecture Software and Interaction Flow Modelling Language for Tourism Data Acquisition in Colombia. Springer WEA”,Communications in Computer and Information Science, Vol. 657. pp. 368-379
- [43] Zuriel, M., Cristina, M., Jose, A.A., Anibal, Z.C.,Carolina, T.B., Sanjay, M., 2016, “A Baseline Domain Specific Language Proposal for Model-Driven Web Engineering Code Generation”, Springer Proc. ICCSA. Lecture Notes in Computer Science,pp. 50-59
- [44] Jose, A.A., Anibal, Z.C., Carolina, T.B., Sanjay, M., Roberto, B., Abraham, O., 2015,“An Analysis of Techniques and Tools for Requirements Elicitation in Model-Driven Web Engineering Methods”,Springer Proc. ICCSA. Lecture Notes in Computer Science, pp. 518-527
- [45] Roberto, A., Aldo, B., Marco, B., Stefano, B., 2015,“Model-Driven Development Based on OMG’s IFML with WebRatio Web and Mobile Platform”, Springer Proc. ICWE.Lecture Notes in Computer Science, pp. 605-608
- [46] Carissa, B.J., Sundaram, N., Wayne, S., Ganesh, A., Shruti, N., 2011,“Modeling web-based information seeking by users who are blind”, Taylor & Francis Journal on Disability and Rehabilitation: Assistive Technology, Vol. 6(6)

- [47] Luis, I., Nicolas, P., Javier, C., Jose, A.A., Rosa, A., 2010, "A Model Transformation Approach for Automatic Composition of COTS User Interfaces in Web-Based Information Systems", Taylor & Francis Journal on Information System Management, Vol. 27(3)
- [48] Mohamed, L., Abdelmounaïm, A., 2016, "Modeling and generating native code for cross-platform mobile applications using DSL", Taylor & Francis Journal on Intelligent Automation & Soft Computing, Vol. 1
- [49] Gustavo, R., 2013, "Web Modeling Languages Strike Back", IEEE Internet Computing, Vol. 17(4)
- [50] Muhammad, R., Muhammad, W.A., Aamir, M.K., 2015, "Towards the Tools Selection in Model Based System Engineering for Embedded Systems - A Systematic Literature Review", Elsevier Journal of Systems and Software, Vol. 106
- [51] Karel, F., Miroslav, B., and Ivan, J., 2015, "Using the Interaction Flow Modelling Language for Generation of Automated Front-End Tests", Federated Conference on Computer Science and Information Systems (ACSIS)
- [52] Karel, F., Miroslav, B., and Ivan, J., 2015, "Transformation of IFML schemas to automated tests", Proceedings of the 2015 Conference on research in adaptive and convergent systems (RACS)
- [53] Eman, M. S., and Omar, A. S. S., 2015, "A Model-Driven Engineering Transition-Based GUI Testing Technique", International Conference on Computational Science and Computational Intelligence
-
- [54] Carlo, B., Sara, C., and Piero, F., 2017, "IFMLEdit.org: Model Driven Rapid Prototyping of Mobile Apps", IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)
- [55] Carlo, B., Sara, C., and Piero, F., 2017, "IFMLEdit.org: a Web Tool for Model Based Rapid Prototyping of Web and Mobile Applications", MISE, Tool Demo
- [56] Carlo, B., Sara, C., and Piero, F., 2017, "Online Model Editing, Simulation and Code Generation for Web and Mobile Applications", IEEE/ACM 9th International Workshop on Modelling in Software Engineering (MiSE)
- [57] Carlo, B., 2017, "ALMOsT.js: An Agile Model to Model and Model to Text Transformation Framework", International Conference on Web Engineering (ICWE), pp.79-97

-
- [58] Rocio, N. T., and Carlo, B., 2017,“ALMOsT-Trace: A Web Based Embeddable Tracing Tool for ALMOsT.js”, International Conference on Web Engineering (ICWE), pp. 554-558
- [59] Judy, B., and Steve, R., 2017,“Generating Obligations, Assertions and Tests from UI Models”, Proceedings of the ACM on Human-Computer Interaction-EICS, Vol. 1(1)
- [60] Priya, G., and Prafullakumar, S., 2011,“Model based Approach to Assist Test Case Creation, Execution, and Maintenance for Test Automation”, Proceedings of the First International Workshop on End-to-End Test Script Engineering ETSE’11, pp. 1-7
- [61] Robert, C., Armstrong, Ratish, J., Punnoose, Matthew, H., Wong, Jackson, R., Mayo, 2014, “Survey of Existing Tools for Formal Verification”, Technical Report SAND2014-20533 551829, 2014
- [62] Muhammad, A. B., Jamil, A., and Fahim, A., 2014, “Modeling of Real-Time Embedded Systems using SysML and its Verification using UPPAAL and DiVinE”, 5th IEEE International Conference on Software Engineering and Service Science (ICSESS)