# A Survey of Analysis Models and Methods in Website Verification and Testing

Technical Report 2007-532

Manar Alalfi, James R.Cordy, and Thomas R. Dean

School of Computing
Queen's University
Kingston, Ontario, Canada
{alalfi,cordy,dean}@cs.queensu.ca

February 2007

## Abstract

Models are considered an essential step in capturing different system behaviors and in simplifying any further analysis required to check or to improve the quality of software. Verification and testing of web software requires effective modelling techniques that address the specific challenges of web applications. In this study we survey 21 different analysis modelling methods used in website verification and testing. Based on our survey, a categorization, comparison and evaluation for such models and methods is provided.

# Contents

# List of Tables



# List of Figures

# 1 Introduction

Web applications are becoming more complex. Complexity arises due to several factors, such as a larger number of hyperlinks, more complex interaction, and increased use of distributed servers. Modelling helps to manage the complexity of these systems. Several papers in the literature have studied the problem of web applications modelling. Different models have been proposed, while others have been adapted from existing modelling techniques for other types of software. Modelling support is essential to provide an abstract view of the application. It can help designers during the design phases by formally defining the requirements, providing multiple levels of detail as well as providing support for testing prior to implementation. Support from modelling can also be used in later phases to support verification.

Most of the early literature concentrates on the process of modelling the design of web applications. It proposes forward engineering-based methodologies designed to simplify the process of building a highly interactive web applications. Other research uses reverse engineering methodologies to extract models from existing web applications in order to support their maintenance and evolution.

This survey studies a range of different analysis models that are currently applied in the field of verification and testing of web applications. Design modelling methodologies are outside the scope of our study. While reviewing different methods, we found that some of literature focuses on modelling the navigational aspects of web applications. Others concentrate on solving problems arising from user interaction with the browser in a way that affects the underlying business process. Still others are interested in dealing with static and dynamic behavior. In this paper, we attempt to categorize these methods according to the level of web application modelling - navigation, behavior, and content. In each category, methods are sorted according to the kind of notation employed by each method. A comparison and evaluation of 21 different methods is described.

The rest of this paper is organized as follows: Section 2 gives a brief introduction into web applications and web services, its then describes the challenges that affect the analysis and modelling of web applications; Section 3 lists the desirable properties of web application modelling and the notations employed by the reviewed modelling methods; Section 4 describes the set of comparison and categorization criteria used in our study. Section 5 gives a brief summary and a comparative analysis of 21 modelling method. Finally, a conclusion and open problems are given in Section 6.

# 2 Website Modelling

## 2.1 Web Applications

A Web application is a software application that is accessible via a thin client (i.e. web browser) over a network such as the Internet or an intranet. A Web application is often

structured as a three-tiered application. The web browser represents the first tier; the web server that uses some dynamic web content technology such as CGI, PHP, Java Servlets or Active Server Pages( ASP ), is the middle tier; and the application server that interacts with the database and other web objects is considered the third tier.

Web applications render web pages, comprising different kinds of information such as text, images, forms. Web pages can be static or dynamic. Static pages reside on a web server that contains only HTML code, and an executable code that runs on the web browser and is served by the web server; however, they do not need to be preprocessed by the application server. Dynamic pages are modifiable, as a result of the execution of various scripts and components at the server. These pages contain a mixture of HTML tags and executable code, and are served by the application server.

Figure 1 shows the main components and elements in the process of web interaction. The basic unit of this interaction is the Web page itself. It is sent to a browser from a Web server, based on a request from that browser. The browser parses the information in the HTML file, and the resulting user interface is displayed within the browser itself. The role of the Web server is to listen for a request from the client, parse the request to determine the page that the client requested and determine if it can fulfill the request directly, or if the application server must be invoked. The web server serves directly static HTML pages; multimedia content such as images, videos, or audio files; or it forwards the request to the application server. The application server preprocesses dynamic (active) pages, integrates data from various resources such as web objects or databases, and then it returns the result to the web server as static HTML pages. The web server in turn returns the HTML page to the requesting web browser, which displays it to the user. At this point, the server forgets everything about sending that file to the client, except for maybe placing an entry into a log file. It is this "connection-less" nature that gives a web server its scalability, but in turn makes it a challenge to create meaningful applications without some additional support.

In our study we consider web application and web site as synonyms. Some researchers consider website as a set of related web pages grouped together by some means, and existing on a server, or within a folder on that server. Such pages are static pages that don't contain any dynamic feature in the sense that they don't have to be processed by the application servers. Here we are interested in all methods that propose models to capture different properties related to the structure(navigation), behavior, and content of web applications; and whether these properties are static, dynamic, or interaction.

## 2.2 Web Services

In this survey we do not consider Web services, which are a standardized way of integrating web-based applications. Web services are primarily used as a means for businesses to communicate with each other and with clients, without intimate knowledge of each other's IT systems. All communication is in XML, and not tied to any operating system or program-

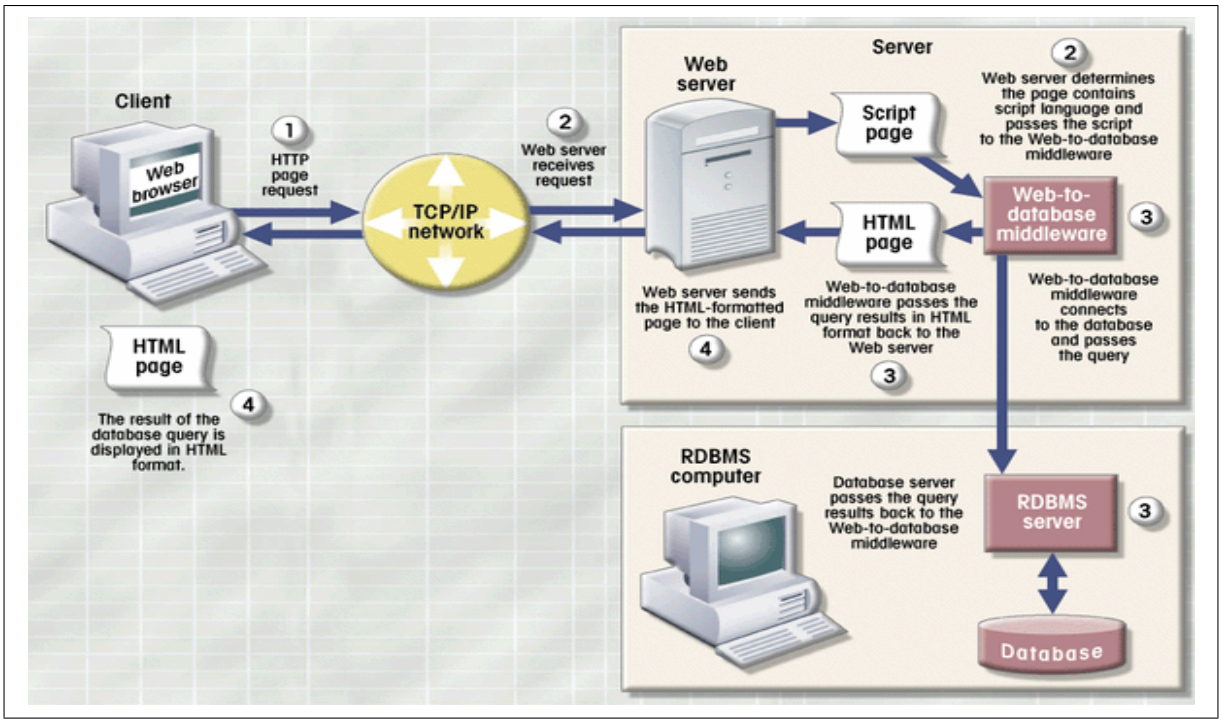


Figure 1: Web Application Components [RC04]

ming language. Web services don't require the use of browsers or HTML, and don't provide the user with a GUI. This class of software is outside the scope of our study, but it may be a future direction for our work.

## 2.3 Challenges in Analysis and Modelling of Websites

Web applications are evolving rapidly, using many new technologies, languages, and programming models that are being used to increase the interactivity and the usability of web applications. This inherent complexity brings challenges to modelling, analysis, testing, verification and maintenance of this kind of software. Some of these challenges are:

- The diversity and complexity of the web application environment increases the risk of non-interoperability and the complexity of integration. Web applications interact with many components that run on diverse hardware and software platforms. They are written in diverse languages; and they are based on different programming approaches such as procedural, OO, interpreted, and hybrid languages like JSP. The client side includes browsers, embedded scripting languages and applets. The server side includes HTML, CGI, Java Server Pages (JSPs), Java Servlets, and .NET technologies. They all interact with diverse back-end engines and other components that are found on the web server or other computers behind the server. The integration of such components and the web system in general is extremely loose and dynamically coupled, which provides

powerful abstraction capabilities to the developers, but makes analysis for testing and verification extremely difficult.

- Another major challenge comes from the dynamic aspects, including dynamically generated client components, dynamic interaction among clients and servers, and the continual changes in the system context and web technologies.

- Web applications may have several entry points, and users can engage in complicated interactions that the web application cannot prevent. Web applications are often interfaced to database systems and provide the same data to different users. In these cases, applying access control mechanisms becomes an important requirement for safe and secure access to web application resources, and the process of implementing and applying such rules is considered a great challenge.

- Some information in web applications is transmitted through hidden fields and special channels, due to the stateless behavior of the HTTP protocol. It's considered a challenge to have a precise analysis for web applications that takes into account this information as well.

# 3 Modelling

## 3.1 Desirable Properties for Website Modelling

We can think of web applications from three orthogonal perspectives: web navigation, web content and web behavior. Desirable properties of web applications can fall within these three dimensions. Desirable properties can be classified into:

- *Static navigation properties.* Most of the early literature on web analysis and modelling concentrates on dealing with static links, treating web applications as hypermedia applications. It addresses checking of properties such as broken links, reachability (e.g., return to the home page), consistency of frame structure, and other features related to estimating the cost of navigation, such as longest path analysis.

- *Dynamic navigation properties.* This kind of analysis focuses on aspects that make the navigation dynamic. That is, the same link may lead to different pages depending on given inputs. The inputs could be user inputs transferred via forms, or system inputs depending on some state in the server such as date, time, session information, access control information or information in hidden fields.

- *Interaction navigation properties.* This kind of analysis focuses on properties that are related to user navigation that happens outside the control of the web application, such as user interaction with the browser. This includes features such as using the back button, the forward button, and URL rewriting.

| Feature modeled or property checked | | Feature or property Description | Example Formula | Formula Type | Reference |
|---|---|---|---|---|---|
| *Static Navigation Properties* | *Broken links* | Absence of broken links in the web site | *Φ1 = AG(link → EX page), for each link in the web site, a page exist that is attached to it in the next sate.* | *CTL* | *[SDMP02]* |
| | *Reachablity* | There is at least one navigation path from the start page to Queue page. | EF (*PageVar = PageName*), for example: EF (Page = Queue). | *CTL* | *[HH06]* |
| | *Dead End* | Along any path from start point it is always possible to reach Queue page. (Queue page is reachable from any other page.) | AG EF (*PageVar = pageName*) for example: AG EF (Page = Queue) | *CTL* | *[HH06]* |
| | *Frames consistency* | – Duplicated frame names (a name l that occurs in more than one frame tag).<br>– Frame trees deeper than a fixed threshold.<br>– Non-existent link targets (anchors tags < a, l > such that l does not appear in any frame tag). [dA01] | [] *p*, where *p = duplicateFrames_mainW = = 0*<br>*duplicateFrames_mainW is a Boolean variable that is set to True if two frames having same name are active simultaneously. This property requires the absence of a frames error where frames having same names are not active simultaneously* | *LTL* | *[HPS04]* |
| | *Form filling* | *The ability of modeling form_based pages, and to populate those forms with different values automatically or semi-automatically.* | | | |
| | *Longest path* | The "length" of a path consists of the number of bytes, or the number of links, that must be downloaded in order to follow it. In MCWEB, there is an extension that enables the computation of the longest and shortest paths in a set of webnodes. | To find the all-pair longest path between webnodes of a domain $_{\Delta}$ MCWEB post-processes the output of the formula, a: home page a; Post(x): webnodes reachable by following one edge from x, in_ domain $_{\Delta}$ : holds for a webnode w if there is an URLpage s in w such that s N contains the substring $_{\Delta}$ ;<br>$\langle\langle \mu x.x = a \cup (Post(x) \cap indomain_{\Delta}) \rangle, x \rangle$<br>The computation of the all-pair longest path can provide information about the bottlenecks in the navigation of a site. | Constructive μ - calculus | [dA01] |
| *Dynamic Navigation* | *System input*<br>*User input* | To model the feature of having the target of the same navigation link determined at run time depending on some conditions provided the by user, or some kind of automated processing done by the system. For example, some links are available if the user has some access rights; search engines such as Google, which depends on user's keywords in generating a document containing dynamically generated links for each document corresponds to the user's keywords. | | | |
| *Interaction Navigation* | *HTML + user operations* | Modeling and checking the user interactions with the browse that may affect the business logic of web application; this could include modeling the back button, the forward button, and URL rewriting. The following sequence of steps generates the Amazon bug, a well known bug caused by ignoring user interactions with the browser.<br>*Step 1*: The shopping cart of the user is empty and the user browses the web site.<br>*Step 2*: The user adds an item *Item1* to the shopping cart.<br>*Step 3*: The user decides that he does not want to buy *Item1* after all, but instead of deleting it from the shopping cart he presses the "back" button to return to the previous shopping cart which is empty. | all s: State \| s.browser.display.cHasItems = s.browser.bl.scHasItems<br>*A major requirement of the model is to guarantee the integrity of the system by ensuring that the list of items that are displayed on the browses current web page (cHasItems) is identical to the contents of the shopping cart (* scHasItems*); bl ( bussniss logic) is an abstract class that relates the browser to it's data content. Using Alloy Analyzer one can see that the assertion fails.* | *Alloy* | *[BA05]* |

Table 1: Desirable Properties for Website Modelling

Table 2: Desirable Properties for Website Modelling (Cont.)

| Feature modeled or property checked | | Feature or property Description | Example Formula | Formula Type | Reference |
|---|---|---|---|---|---|
| *Static content properties* | *Incomplete WP* | The model should enforce that some information is available at a given Web page, some links between pages do exist or even the existence of the Web pages themselves.<br><br>L→#r :If L is recognized in some web page of W, then r must be recognized in some web page of W which contain the marked part of r. | member(name(X), surname(Y )) → hpage(name(X), surname(Y ), status())<br><br>*If there is a Web page containing a member list, then for each member, a home page exists containing (at least) the name, the surname and the status of this member.* | Rewriting-based specification language | *[ABF05]* |
| | *Incorrect WP* | L →error \| C<br>If L is recognized in some web page of W and all the expressions represented in C are evaluated to True (or C is empty), the web page is incorrect. | project(year(X)) → error \| X in[0 − 9]*, X < 1990<br><br>*If there is a Web page containing a project year, where the year is numeric and less than 1990, it should be replaced with error.* | Rewriting-based specification language | *[ABF05]* |
| *Dynamic content properties* | *Incomplete WP* | To be able to check the syntax and semantics (specifically the incomplete property) of dynamically generated content that is resulting from the execution of scripts by the application server.<br>*None of the modeling methods did such kind of checking.* | | | |
| | *Incorrect WP* | To be able to check the syntax and semantics (specifically the incorrect property) of dynamically generated content that is resulting from the execution of scripts by the application server.<br>*None of the modeling methods did such kind of checking.* | | | |
| | *New connection* | *To be able to model connection which it's source and target is determined by the system at run time. For example in an electronic book which has 200 chapters; linking each one individually in the content list is time consuming; time could be saved by using an algorithm that can use user selected text (in the content list) to automatically links the chapter with a title corresponding to the selected text.* | | | |
| | *New content* | *To be able to model new generated components that the user can't determine until run time.* | | | |
| *Instruction processing* | *Server-side execution* | *If the method is providing a model for the code that is being executed on the client or the server side, and whether the method is able to specify the location of such execution; is it happening on the client or the server side.* | | | |
| | *Client-side execution* | | | | |
| *Security properties* | *Access control* | A member cannot have administrator functions and an anonymous user cannot view pages belonging to a member | AG(member → !all); AG(noLog → (!partialΛ!all))<br>*All: administrator functions; partial: member functions.* | *CTL* | *[CMRT06]* |
| | *Session/cookie.* | To check whether or not the inactive period of the current session is over a time limit, say, 5 minutes. If the inactive period is greater than the time limit, the HTTP request must be redirected to an authentication page to verify the user again for protecting the user's privacy. | $(\exists s \in Session)((s = this.Session() \wedge s.Inactive() > 5) \rightarrow this.redirect(Auth))$<br><br>*The this.Session ( ) is a function that retums a session object ; s.1nactive() is a function that returns the inactive period for the session object s. The this.redirect (Auth) specifies that the HTTP request is redirected to the Auth server-page.* | *First-order logic* | *[KLH00]* |

- *Static content properties.* Consistency of the web page content with respect to syntax and semantics.

- *Dynamic content properties.* This analysis requires the ability to check the syntax and semantics of dynamically generated content that results from the execution of scripts by the application server. Some technologies are able also to generate new connections, some of which may be to a remote site. Also new web components could be generated at run time, and these components must also be analyzed.

- *Security properties.* This issue is related to access control mechanisms that could be employed on the web content or web links. This issue could also be employed on the backend, as the database contains data reserved to specific users; non-authorized users can not access such data. These properties are also tied to session control mechanisms.

- *Instruction processing properties.* This includes server and client side execution. We can define client-side execution as any process changing the state of the application without communication with the web server. Server-side execution is defined by all instructions processed on a web server in response to a client's request. A modeling method should to be able to model these features and to recognize whether execution is done on the server or on the client.

Table 1 and 2 provides an example for each of the above properties.

## 3.2 Notation Employed by the Reviewed Modelling Methods

- ***StateCharts***

  StateCharts [Har87] is a visual formalism that extends state diagrams for modelling complex/reactive systems. StateCharts can be defined as a set of: states; transitions; events; conditions and variables and their inter-relations. StateCharts propose three basic structuring mechanisms extending the flat nature of state diagrams and increasing its expressiveness, understandability and maintainability. The first two are critical for web applications navigation modelling:

  1. Hierarchy; is represented by the composition of nested states (XOR-states) thus allowing an efficient use of transition arrows. XOR-states can have exclusively one active sub-state at a time.
  2. Orthogonality; is the decomposition of composite states into concurrent regions representing independent modules in a system. Each concurrent region in an AND-state is delimited by a dashed row. Like a XOR-state, each concurrent region can have at most one active state at a time.
  3. Broadcasting communication; is represented by events that are associated with more than one transition. In that case, when an event happens, all transitions

associated with the triggered event are evaluated and executed if the guarding conditions are true. In classical statecharts, activities and events are considered to be instantaneous (they take no time to perform).

States in StateCharts can be initial, final and standard states. A standard state in turn can be classified as a simple, orthogonal or hierarchal state. The operational semantics for StateCharts are given by a sequence of steps. At each step the state machine evaluates a single transition and may assume a new state configuration (the set of currently active states). When an event happens, the system transmits it to the transition associated to the triggered event. Then, the corresponding guard condition is evaluated, and if it is true the state machine sets the target state as active. An optional activity can be added to the transition label, indicating which activity will take place when the transition happens.

- ***Labelled Transition Systems***

Labelled transition [Wik]is a quadruple $(State, Label, \rightarrow, s_0)$ where (i) State is the set of possible states of the program computation; (ii) Label is a set of labels showing the information about the state changes; (iii) $\rightarrow \subseteq$ State $\times$ Label $\times$ State is a transition relation that describes the system evolution. (s,l,$\acute{s}$)$\in\rightarrow$ expresses that the system evolves from state s to state $\acute{s}$ with the information of the state change described in l, (iv), $s_0 \in$ State is the initial state.

- ***SDL***

Specification and Description Language (SDL) is a specification language targeted at the unambiguous specification and description of the behavior of reactive and distributed systems. Our description will focus on two aspects [FHvLP00].

  - Agent is an extended finite communication state machine that has its own identity, its own signal input queue, its own life cycle and a reactive behavior specification. Three main parts exist on agent declarations: attributes, behavior, and internal structure. Agents are composed of systems, blocks and processes. Processes are used to specify the behavior of the system. Variables store data local to an agent and they are owned by the agent's state machine. They can be private, local, or public. System is the entry point of the SDL specification. It includes a set of blocks and channels. Blocks are connected to each other and to environment by channels. A block can contain either a set of processes or a set of block substructures. A state machine is either in a state waiting for a signal or performing a transition. A transition results in entering a new state or stopping the agent.
  - Communication is based on signal exchanges that carry the signal name, user data and sender identification. It requires a complete path from sender to receiver

that consists of channels, gates and connections. SDL processes communicate with each other and with the environment by exchanging signals. Each process instance in SDL owns a dedicated input port that allows received signals to be queued until they are consumed or discarded by the process instance owner.

SDL supports modelling of non-determinism for transitions, postponing signals, and condition signals. A Message Sequence Chart (MSC) describes a specific execution sequence of the system. It shows how messages are passed between the instances. Instances are described by vertical lines that define the temporal ordering of the events related to the instance. An instance represents an SDL block, process or service. Messages represent the interaction between instances or between an instance and the environment. The message is described by a horizontal arrow. Conditions in MSC represent a notion of state, which is represented by a hexagon. An action in MSC is represented by a rectangle and it describes an internal activity of the instance.

- ***UML and OCL***

  UML [Wik] is a family of languages that is mainly used in the modelling and specification of object-oriented systems. The UML defines a number of diagrams, some of which depict the static structure of a system, while others the dynamic aspect. For example UML class diagrams model the static structure of a system. Where the behavior of the system can be described by OCL. OCL is a textual language that adds formalism to UML diagrams. It can be used to define the behavior of a model (with the use of preconditions and postconditions) or to express constraints (using invariants) on the elements of a UML model.

- ***UML-based Web Engineering(UWE)***

  The UWE meta-model is designed as a conservative extension of the UML meta-model (version 1.4). Conservative means that the modeling elements of the UML meta-model are not modified e.g. by adding additional features or associations to the modeling element Class. All new modeling elements of the UWE meta-model are related by inheritance to at least one modeling element of the UML meta-model. The authors define for them additional features and relationships to other meta-model modeling elements and use OCL constraints to specify the additional static semantics[KK03].

  In UWE the content, navigation, presentation, and business process of web application are recognized as separate concerns and modelled separately. The content of web applications is modeled in a conceptual model where the objects that will be used in the Web application are represented by instances of $\langle\langle$ conceptual class $\rangle\rangle$ which is a subclass of the UML Class. Relationships between the objects are modelled by UML associations between conceptual classes. The navigational path is modelled using a navigational model, where each navigational node is represented by a $\langle\langle$ navigational class $\rangle\rangle$ and is associated to a conceptual class containing the information of the node.

Navigational nodes are linked by associations. The business process logic of web application is represented by a process model using the UML activity diagram where the presentation model is used to sketch the layout of the web pages associated to the navigational nodes.

- ***Alloy***

  Alloy [Jac02] is a textual modelling language based on first order predicate logic. Entities in the model are described by Signatures which are similar to a UML Class diagram. Signature can define fields, which are like attributes of classes in UML class diagrams. Facts, Predicate and Assertion are three kinds of expressions defined by Alloy model. A Fact is an expression that every instance of a model satisfies. Predicates and Functions are like functions in an object-oriented programming language. They can be invoked from other parts of the model. Finally, an assertion is a statement that the modeller wants to check for its validity. Alloy models can be analyzed by Alloy Analyzer; counterexample is being generated by the analyzer if an assertion is violated. State explosion problem is being controlled by introducing scoped analysis. A scope determines the maximum number of instances the Alloy Analyzer probes to ensure the validity of an assertion or to find the existence of counterexamples. If the Alloy Analyzer fails to come up with a counterexample, the assertion may still be valid. The bigger the scope is, the more confident the modeller is that his or her model is correct.

- ***Directed Graph and CFG***
  A directed graph or digraph G [Wik] is an ordered pair G: = (N,E) with

  - N, a set of vertices or nodes,
  - E, a set of ordered pairs of vertices, called directed edges, arcs, or arrows.

  Many methods for web site analysis and verification use directed graph to model the structure of the web site, where N represents a set of web pages and E represents a set of links between the pages; this model is called Flat model.

  The logical structure of the web application is represented by many static analysis-based methods using control flow graph (CFG). Nodes in this graph represent computational statements or expressions, the edges represent transfer of control between nodes, and each possible execution path of the module has a corresponding path from the entry to the exit node of the graph.

- ***FSM***
  A finite state machine (FSM) or finite state automata [Wik] is a model of behavior composed of a finite number of states, a start state, an input alphabet, and a transition function that maps input symbols and current states to a next state, FSM can be deterministic finite state machine (DFA) when it has at most one transition for each symbol and state, while nondeterministic finite state machine (NDFA) transitions are conditioned on no input symbol (a null) or more than one transition for a given symbol

and state. When NFA states are labeled with Boolean variables, which are the evaluations of expressions in that state, and it may be extended with fairness constraints it called Kripke structure. Finite state machines are widely used in modelling of application behavior, design of hardware digital systems, software engineering, compilers, network protocols, and the study of computation and languages.

- ***Rewriting Systems***
  A term rewriting system (TRS for short) [Wik] consists of a set of function symbols and a set of rewriting rules in the form of l → r, where l and r are first order terms, left-hand-side l is not a variable and, each variable occurring in r also occurs in l. The execution of a rewrite system involves the repeated application of the rules to some context. In each application, an occurrence of the left-hand side of a rule in the context is replaced by the right-hand side. The execution terminates when no matching rule can be found anymore. For example the following TRS defines multiplication over natural numbers.

  a(0, y) → y
  a(s(x), y) → s(a(x, y))

  m(0, y)→ 0
  m(s(x), y) → a(y, m(x, y))

  Here, a stands for addition and m stands for multiplication.

# 4 Comparison and Categorization Criteria

## 4.1 Feature Type

We note the web application features that are being captured by the proposed models, and the properties that the modeling methods are capable of checking. These features are categorized into static, dynamic and interaction features:

- *Static Features.* This includes static properties of web applications. It is mainly concerned with links that connect an HTML page with other HTML pages. When the user clicks on a static button or a static link, a request is sent to the server in order to fetch a page. The server responds to the request by retrieving the required page from its storage and sends it back to the client. In this scenario several properties can be checked; these are related to static navigation and static content properties.

- *Dynamic Features.* These features include dynamic links and dynamic content properties. Dynamic links describe the connection between HTML pages and code that must to be executed on the server in order to generate the required information, build it into an HTML page, and return it to the client. The processing done by the server may depend on input that is provided by the user or the system. User inputs are usually

sent by filling a form or by hidden fields in the HTTP request. System inputs depend on the server state, such as server time, or on some kind of interaction with other resources, such as database servers or web objects. The output could be constructed as new content, or a link in a new HTML page. Properties that fall into this category are those related to dynamic navigation properties, dynamic content properties, security properties, and instruction processing properties.

- *Interaction Features.* This includes properties related to user interaction with the browser. The browser's influence on the navigation behavior of the web applications should be taken into consideration while modelling or analyzing web applications, as the web browser provides the interface to the web applications, and can change the navigation behavior while a user browses a web application.

## 4.2 Notation

Modeling methods use different notations; some of them are formal, while others are either semi- or informal. The main notations that are used by the methods reviewed in our study are described in section 3.2.

## 4.3 Level of Modelling

Web application modelling can be viewed from different perspectives. We compare the modelling methods here according to three basic levels: content, structure (navigation), and behavior. These three levels in turn could have a static or a dynamic flavor.

## 4.4 Application of the Model

In our study we focus on methods that are concerned with modelling web applications for the purpose of testing or verification; this also could include design verification.

## 4.5 Is Source code Required?

Modeling methods may require doing a white-box or a black-box analysis. This determines whether or not the existence of the source code is required for the analysis. The kind of analysis for each reviewed method is specified.

## 4.6 Model Optimization

Complex systems in general have a state explosion problem or they generate a large complex model. In all cases such models need some sort of optimization. In web applications, this problem becomes a major challenge to the success of any method that attempts to analyze and model a scalable web system.

### 4.7 Tool Support

We list if the method being described is supported either with a proposed tool, or with a preexisting tool.

## 5 Survey and Comparison Results

### 5.1 Survey

Our study resulted in two different views of the methods we surveyed, a general categorization by modelling level, and a detailed comparison by property coverage.Table 5 summarizes the first one, where the 21 methods are categorized according to the level of web application modelling, as interaction behavior modelling methods, navigation modelling methods, content modelling methods and hybrid modelling methods (methods that model more than one level).In each category, methods are sorted according to the notation used by the method. At the same time, comparison between the methods was also done based on other criteria such as: application for the method (analysis, testing, verification or some combination); whether the source code is required for the analysis or not; the way the method solves the state space explosion problem; and finally, whether there is tool support for the method.

| Method Name | Feature type | Notation | Level | Application | Source code required | Model optimization | Tool support |
|---|---|---|---|---|---|---|---|
| [Con99 ] | Static | Extended UML | Structure (Navigation) | Analysis | No | No | Rational Rose Tools |
| [RT00] | Static | Directed graph | Structure (Navigation) | Analysis + can be use for verification & testing | Yes | No | ReWeb |
| [dA01] and [dAHM01] | Static | Directed graph With Web Nodes | Structure (Navigation) | Verification | No | No | MCWeb |
| [ABF05] | Static | Partial rewriting | Content | WS verification Tool(GVerdi) | Yes | No | GVerdi prototype |
| [GFKF03] | Interaction | Abstract model, use lambda calculus | Interaction Behavior | WA interaction with the browser | No | No | prototype |
| [LK04] | Interaction | WebCFG | Interaction Behavior | Verification | Yes | Yes | Implement a model checker |
| [CZ04] | Interaction +Static | Labeled transition | Interaction + static (Navigations) | Testing and verification | No | Yes | None |
| [BA05] | Interaction | UML(WS structure) OCL( behavior of the model) | Interaction Behavior | Verification for user interaction( Amazon + Orbitz bug) | No | Yes | UML2Alloy |
| [WP03] | Static + dynamic | Extended StateCharts | Navigation | Design verification | Yes | Yes | SWCEditor |
| [HH06] FARNav | Static + dynamic | StateC harts | Adaptive Navegation | design and implementation Verification + testing | No | Yes | Existing SVM model-checking tools |
| [KLH00] WTM | Static + dynamic | Control flow graph, data flow graph, and finite state machines OSD( object state diagram) | Static and dynamic Behavior, Dynamic Navigation | Testing | Yes | No | None |
| [HPS04] | Static+ dynamic | System of communicating automata | Navigation + Behavior | WA Verification | No | Yes | Fame Work with GUI + network monitoring tool + analysis tool |
| [AOA05] FSMWeb | static + dynamic | hierarchies of Finite State Machines (FSM) | Navigation + Behavior | System level testing | No | Yes | Prototype |
| [BFG02] Veriweb | static + dynamic | Directed graph | Navigation + Behavior | WS testing | Yes | Yes | VeriSoft + web Navigator + ChoiceFinder + SmartProfiles |
| [SDMP02] | Static + dynamic | Web graph | Structure (Navigation) | Verification | No | No | AnWeb prototype |
| [SDM+05] and [CMRT06] | Static + dynamic | WA graph + extension to Kripke structure | Structure (Navigation) | WA design Verification | No | No | WAVer + SMV tools |
| [TR04] And [TR02] | Static + dynamic | (model navigation layer) + CFG (client & server code) | Structure (Navigation) + Behavior | Testing | Yes | No | ReWeb + TestWeb |
| [BMT04] | Static+ dynamic | UML-meta Model + UML state diagram | Structure (Navigation) | Analysis & Testing | Yes | No | WebUML prototype |
| [KZ06] | Static + dynamic | Extended UML (UWE) | Navigation + Behavior | Design Validation and Verification | No | No | ArgoUWE + Spin or UPPAAL |
| [WO02] | Interaction + static + dynamic | Regular expression | Interaction + dynamic Behavior | Can be used for testing + implementation + impact analysis | Yes | No | None |
| [SM03] | Static + dynamic | SDL | Structure (Navigation) | Testing and verification | Yes | No | Existing SDL Support tool |



Table 3: Survey Result (categorized according to feature type)

### 5.1.1 Static Modelling Methods

1. **Notation**: Extended UML

   **Conallen** extends UML to represent the additional features of WA [Con99]. The aim of this work is helping designers, implementers, and architects to integrate modeling web specific elements with the rest of the application model in a coherent and complete way. In his approach, Conallen performs such extension by proposing new constructs to UML, such as Stereotypes, Tagged values and Constraints. Stereotypes in UML allow the definition of new semantics for a modeling element; Tagged values represents new properties that can be associated to model elements; Constraints specify new conditions under which a model can be considered "well-formed". Stereotype is used to define two types of web pages, client page and server page, where a web page is modeled basically as a class with the semantics of client and server page defined by stereotype. The relation between Server page and client page is defined using the stereotype ⟨⟨build ⟩⟩, because the client page is usually created by the server. The relationship between web pages is defined using the stereotype ⟨⟨ link ⟩⟩. Moreover The HTML element, such as JavaScript, Java applets, AciveX controls, form, or frame, is considered also as a stereotyped class. Tagged values are used to define the parameters that are passed along with a link request. The "link" association tagged value "Parameters" is a list of parameter names (and optional values) that are expected and used by the server page that processes the request. Although this approach helps in modeling all web application static and dynamic features, no specific modeling method for any phase of web application is described. the Conallen model is considered the basis of many analysis modeling methods.

   **Features modelled:** client pages, server pages, hyperlinks, forms, frames, JavaScript, Java applets, AciveX controls.

   **Tool support:** Rational Rose tools.

2. **Notation**: Directed graph.

   **Ricca**, in his PhD thesis adapts an approach to analyze, test, and restructure web application based on a reverse engineering paradigm [Ric04]. He didn't propose models and formalisms to support the design of web applications; instead, based on the assumption that a web application already exists, he investigates different well established methods for the analysis, testing and re-structuring of traditional software systems, adapting them to the case of Web applications. In [RT00] web application is modelled as a graph; nodes and edges are split into different subsets. Nodes subsets are a set of all web pages; a set of frames for one web page; and a set of all frames. Edges are also split into three subsets according to the kind of target node; a set of hyperlinks between pages or a relation showing the composition of web page into frames;

a set of the relations between frames and pages; as they show which page in which frame is loaded; and a set of relations showing the loading of a page into a particular frame. The name of the frame is given as a label next to the link. This model is implemented in ReWeb. The ReWeb [RT01b] tool consists of three modules: a Spider, an Analyzer and a Viewer. The Spider downloads all pages of a target web site, starting from a given URL and providing the input required by dynamic pages, and then it builds a model of the downloaded site. The Analyzer uses the UML model of the web site and the downloaded pages to perform several analyses. Since the structure of a Web application can be modelled with a graph, as a reinterpretation of the UML model, several known analysis, working on graphs, such as flow analysis and traversal algorithms can be applied. The Viewer provides a Graphical User Interface (GUI) to display the Web application view as well as the textual output (reports) of the analyses.

**Tool support**: ReWeb.

**Features modelled**: ReWeb checks for Static Navigation properties (broken links, reachablity, frames consistency), general graph analysis properties such as longest path, also ReWeb check for cloned pages, which happened When the HTML structure of a page is replicated.

3. **Notation**: Extended Directed graph (Webgraph).

**de Alfaro et al.** model web as a graph with webnodes for vertices, they call it webgraph [dA01], [dAHM01]. A webnode is a hierarchical frame structure, generated by the grammar webnode::= URLpage(name webnode)*. Where an URLpage is the result of fetching a given URL from the Web with a GET method, and each pair (name webnode) consists of the name of a subframe, and of the subframe content. The edges of the graph correspond to links between web pages; the destination webnode is obtained by updating the frame structure as specified by the HTML standard. Taking webnodes, rather than URLpages as vertices of the graph enables an accurate representation of the frame structure of pages. Moreover, since webnodes correspond to pages as displayed by a browser, they lead to a natural connectivity analysis of the web. de Alfaro [dA01] verifies properties written in a slightly restricted $\mu-calculus$ over that model. This technique allows checking many path properties over static web sites such as the password-page property(there is no access to secure pages until the user is signed in), and to present errors as paths through the web model that violate a given property.

**Tool support**: MCWeb.

**Features modelled**: Check for static navigation proprieties(broken links, reachability, frame consistancy), general graph analysis properties such as cost-of-traversal.

4. **Notation:** Partial rewriting.

   **Alpuente et al.** in [ABF05] propose a methodology to verify web sites w.r.t. syntactic as well as semantic properties, GVerdi –the graphical evolution of the VERDI system– implements a rule based specification language for specifying properties of web sites, its a verification technique for automatically checking whether the conditions are fulfilled, and helping to repair faulty web sites; it allows to detect incorrect and incomplete/missing web pages. In this method, web page are modeled as a ground term. Consequently, a web Site is defined as a finite collection of ground terms of a suitable term algebra, a web specification is a triple (R, IN, IM) where

   - R is a set of user's function definitions
   - IN is a set of correctNess rules
   - IM is a set of coMpleteness rules

   Correctness Rules:

   $$l \rightarrow error \shortmid C$$

   if l is recognized in some web page of W and all the expressions represented in C are evaluated to True (or C is empty), the web page is incorrect.
   e.g. $project(year(X)) \rightarrow error \shortmid Xin[0-9]^*, X < 1990$.

   Completeness Rules:

   $$l \rightarrow \sharp r$$

   if l is recognized in some web page of W, then r must be recognized in some web page of W which contain the marked part of r.
   e.g. $member(name(X), surname(Y)) \rightarrow hpage(name(X), surname(Y), status())$.

   This method is being applied only for static web sites, and suitable for recognizing patterns inside semi-structured documents. GVERDI includes a parser for semi-structured expressions and Web specifications, and several modules implementing the graphical user interface, the partial rewriting mechanism and the verification technique. The system allows the user to load a Web site directory together with a Web specification. Additionally, $he\backslash she$ can inspect the loaded data and finally check the Web pages w.r.t the Web site specification.

   **Tool support:** GVERDI (Graphical VErification and Rewriting for Debugging Internet Sites)
   **Features modelled:** (consistency properties) Incorrect + incomplete web pages

### 5.1.2 Interaction Modelling Methods

The abilities of the user to click on the back button in the browser, or to clone a window and submit a request from each clone, or to interact with any browser button or functionality, are called user interactions or user operations. Two well known bugs are generated as a result of this problematic behavior, the Orbitz and the Amazon bugs. The Orbitz Bug is generated according to a sequence of user operations that exposes an actual bug in the flight-reservation program of Orbitz.com [LK04] as described bellow:

**Step 1** a user enters the desired dates and destination of his flight; he is then presented with a page listing possible flights, including Flight A and Flight B.

**Step 2** he clicks a link to open the description of Flight A in a new browser window.

**Step 3** he was not particularly enthused about that flight, he returns to the list of flights, and clicks a link to load the description of Flight B, again in a new browser window.

**Step 4** he decide that Flight A was better after all, he switches back to the window still on the screen showing Flight A, and submits the form, causing a page confirming his reservation to be displayed. The result is Orbitz incorrectly makes a reservation on Flight B.

The Orbitz property asserts the absence of this bug: the flight described on the page that the user submits in Step 4 (which is called the flight-displayed) should be the same as the actual flight for which his reservation is made (the flight-reserved).

Amazon property [LK04], which is drawn from a desired property of Amazon.com: once the user selects a book to purchase, it should be contained in his shopping cart. In particular, the user should be able to select books in two different browser windows and have both appear in his cart, but this means that the cart will not also satisfy the Orbitz property. Researchers are aware of this kind of problem and try to include user operations in their proposed web application models.

5. **Notation:** Abstract Model (Lambda calculus)

**Graunke et al.** devise a model that is dedicated to detect data inconsistency problems, such as the Orbitz bug [GFKF03]. However, these inconsistency problems are only detected dynamically through changes to the server's run-time system. The model also statically discovers abuses of the values filled into form fields. These bugs are a result of user interaction with the browser, or with the application. The presented model has four characteristics. First, it consists of a single server and a single client, because the authors wish to study the problems of sequential Web interactions. Second, it deals exclusively with dynamically generated web pages, called forms, to mirror HTML's sub-language of requests. Third, the model allows the consumer to switch among web pages arbitrarily; this suffices to represent the problem in Orbitz bug and

similar phenomena. Finally, the model is abstract with respect to the programming language so that different alternatives can be experimented on. In this paper lambda calculus was used for forms and basic data, though the authors consider that Classic Java also could be used. Rewriting rules on web configurations is being used to reflect interaction behavior these rules are:$fillform, switch, submit$.Figure 2 describe the proposed model. The proposed model lacks several properties such as ignoring

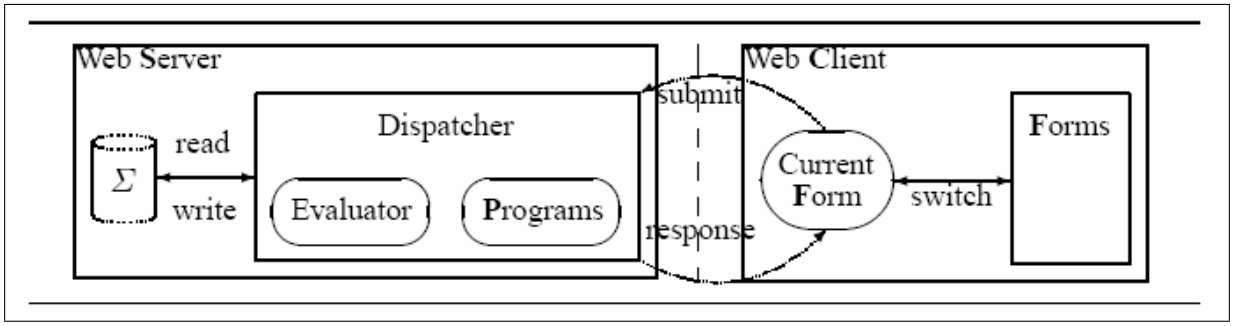


Figure 2: the web picture as described in [GFKF03]

client side storage such as "cookies"; not addressing any security concern; and not addressing concurrency problems, as they just study sequential web interaction.

**Tool support:** the authors implement a prototype for their method.

**Features modelled:** interaction between browsers and the business logic, abuse of values filled into form field.

6. **Notation:** Abstract Model in [GFKF03] + WebCFG (augmented control-flow graph)

**Licata and Krishnamurthi** in [LK04] describe a model checker designed to identify errors in web software. A technique for automatically generating novel models of web programs from their source code was presented. These models include the additional control flow enabled by user operations. In this technique, the authors exploit a constraint-based approach to avoid over-approximating this control flow and to reduce the size of the model. Further, they presented a powerful base property language that permits specification of useful web properties, along with several property idioms that simplify specification of the most common web properties. The authors model a web program P by its web control-flow graph (WebCFG). The WebCFG is an augmented control-flow graph (CFG). User interactions control flow are being added to the model to build a sound verification tool. The authors reduce user operations to primitive user operations proposed by Graunke et al.[GFKF03]. All traditional browser operations can be expressed in this calculus, they just account for switch and submit. Then they construct the WebCFG completely automatically from the source of a web program

using a standard CFG construction technique followed by a simple graph traversal to add the post-web-interaction nodes and the web-interaction edges. The resulting model and properties are checkable by language containment, they are compatible with existing algorithms that supports "constraint" automata, the authors can automatically generate constraints that rule out all the non-infeasible forward paths. This work doesn't address the concurrency issues resulting from multiple simultaneous accesses to a server by different clients.

**Model optimization:** remove some WebCFG states that are not labeled without affecting results, where they use such technique on a web application case study, and were able to reduce state number from 17,000 to 300 states.

**Tool support:** the authors implement their own model checker.

**Features modelled:** proving properties of interactive web sites by discovering user-operation-related bugs, as well as providing a method for verifying all-paths properties of interactive web sites.

7. **Notation: Labelled transition system**

**Chen and Zhao** in [CZ04] use labelled transition to model web system, depending on the conceptual architecture of a web browser as shown in Figure 3. They model the user's interactions with web browsers; the history stack and its impact on the navigation; the local cache and its influence on the freshness of the web pages; and the authentication sessions. The proposed model is described as follows:

The Integrated Model: states

- a page ID to denote the current page - an additional error page err: reached for example, when attempting to access a secure page without an open session.
- a history stack variable for the current status of the URLs contained in history stack. Since in our case there is a one-to-one relationship between a URL and a page id, it is a stack of page ID's
- a variable of a set of page ID's to denote the current status of the locally cached pages.
- a Boolean variable to denote whether the authentication session is currently open.

The Integrated Model: labels L = {(l,f) ∣ l∈ A ∪ {back,forward,entry,err},f=fresh or cache}

- entry: user types in the URL
- back, forward, entry are from browser's interface
- err: navigation is directed to a special error page err

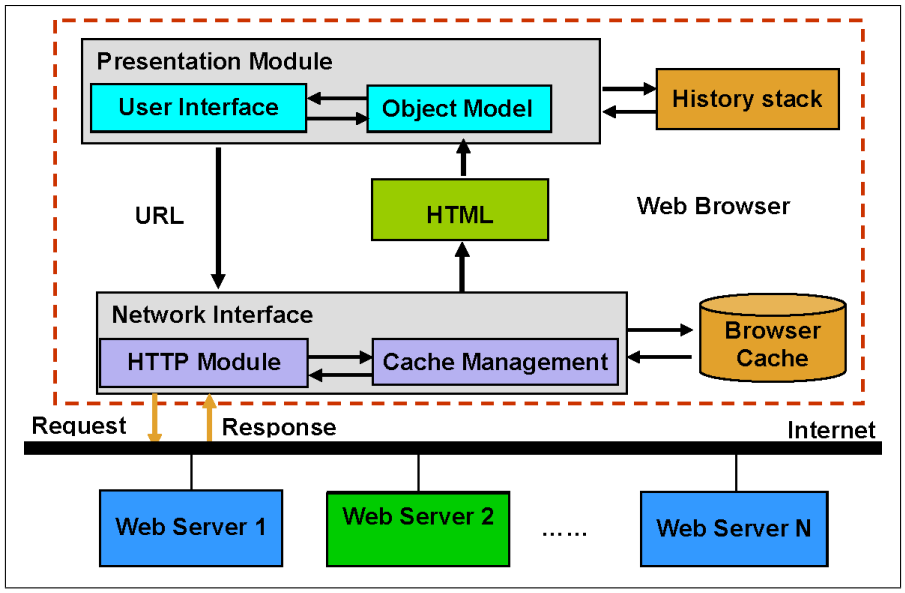


Figure 3: Conceptual architecture of web browser as described in [CZ04]

- fresh/cache: whether the accessed page is from origin server or from local cache.
- A: represent a finite set of symbols for user's actions including SignIn and SignOut

Now given a page navigation diagram $(P, EP, SP, CP, A, \Rightarrow)$,where P, EP, SP, CP are the finite sets of ID's of the pages, entry pages, secure pages and cashable pages respectively; A to represent a finite set of symbols for user's actions including SignIn and SignOut, *and* $\Rightarrow \in P \cup A \cup P$ represent the navigation relation. Let HS(P) denote the set of history stacks of P. Then the labeled transition system is a quadruple $(S, L, \rightarrow, S_0)$, where

- $S \subseteq (P \cup err) \times HS(P \cup err) \times 2^P \times Boolean$;
$-$ $S_0 \in$ S is the initial state;
- $L \subseteq (A \bigcup \{back, forward, entry, err)\} \times \{fresh, cache\}$

The transition relation $\rightarrow$ is defined as the least relation satisfying 20 structural rules. an example for a transition rule of user sign in action:

$$\frac{(p,SignIn,q) \in \rightarrow \bigwedge inChache(lc,q)=true}{\langle p,hs,lc,guard\rangle \xrightarrow{(SignIn,cache)} \langle q,push(hs,q),lc,true\rangle}$$

If the user can sign-in from page p into page q and q is in the cache, then there is a transition from the current state p to the one with page q, where q is put into the history stack. In the ending state, the guard is set true to indicate that the session for authentication is now open. The label on the transition shows that this is a sign-in action.

**State space reduction:** by rendering a reasonable model, abstracting away as much un-related detail as possible, they considered only the navigation behavior influenced by session control and browser caching mechanism.

**Features modelled:** user's interactions with web browsers, the history stack and its impact on the navigation, the local cache and its influence on the freshness of the web pages, and the authentication sessions. Dynamic links are not being considered in the assumed page navigation diagram, and for the functionality provided by session/-cookie techniques, the authors have chosen only the session control.

8. **Notation:** UML ( ADI + OCL)+ Alloy.

**Bordbar and Anastasakis** in [BA05] are interested in identifying bugs such as the Amazon bug and the Orbitz bug, which are created as a result of the interaction between browsers and the business logic. The method makes use of UML class diagrams to model the static structure of a web application, where the behavior of the system is being described via OCL. To analyze the model, the PIM (platform independent model) of web application has to be refined and abstracted to create a new PIM, which is called Abstract Description of Interaction (ADI). The ADI is a class diagram with a set of OCL constraints, and pre- and post-condition expressions, that describe the interaction of the browser and the business logic in an abstract way. The ADI model can be translated to a model in Alloy and analyzed by the Alloy Analyzer. The authors have implemented the transformation from the UML to Alloy in a tool called UML2Alloy, the transformation process starts by creating a UML model of the system in a UML CASE tool, such as ArgoUML, then exporting the UML model to an XMI format, XMI –which stands for XML Metadata Interchange– is an OMG standard used by most UML tools to store, import and export UML models. UML2Alloy implements the transformation and generates an Alloy model from the XMI file. The Alloy model of the system can then be analyzed with the Alloy Analyzer. This method requires a manual effort to construct the ADI model from the PIM; the construction process needs the projection of the PIM, and the deletion of the unrelated model elements.

**Model optimization:** The authors address the problem of state space explosion, by creating an abstract view for the part of the model of the system, which depicts the interaction between the user and the business logic. They also can use state space reduction techniques in Alloy analyzer (scoped analysis).

**Tool support:** UML2Alloy, where they intended to create a tool to generate the ADI by semi-automated methods, using refactoring methods.

**Features modelled:** interaction between browsers and the business logic.

### 5.1.3 Static and Dynamic Modelling Methods(Hybrid)

9. **Notation: StateWebCharts (extended StateCharts)**

   **Winckler and Palanque** in [WP03] extend statecharts to the StateWebCharts (SWC) notation which provides dedicated constructs for modelling specificities of states and transitions in web applications. Their aim is to provide a visual notation easy-to-apply for web designers, and formal enough to be subject of automatic verification, thus supporting designers' activity throughout the design process, most elements included in SWC notation aim at providing explicit feedback about the interaction between users and the system. Currently, SWC is mainly used to describe the navigation between documents rather than interaction between objects. One of the contributions of the proposed SWC notation is that, it makes explicit in the models the points where users interact with the application, with respect to those where the system drives and controls the navigation.

   **Model optimization:** SWC takes benefit from the multilevel hierarchy of classical StateCharts to better manage large web applications.

   **Tool support:** SWCEditor

   **Features modelled:** navigational modelling, web link type support, user driven navigation, system driven navigation, dynamic content generation, frames.

10. **Notation: StateCharts**

    **Han and Hofmeister** present FARNav (a Formal Approach for Rich Navigation modelling) approach [HH06], that uses StateCharts to formally model adaptive navigation, and show how important properties of a navigation model are verified using existing model-checking tool (SMV). StateChart is converted into CTL (input language for SMV). To model adaptive navigation, the authors use parallel (ANDed) substates. The main substate contains a state machine having one state per web page, and transitions between pages for the navigation links. When a web application has only simple (no adaptive) navigation, this substate comprises the entire navigation model. The state machine for the Page Navigation substate is created as follows:

    State: Each web page is a separate state. It is up to the developer to determine what constitutes a distinct web page in an application. Transition: When the user can reach one page from another, there is a transition between the corresponding states. The transition has a label with the form: event [guard]/action; only the event

is mandatory. It describes a user action that causes a request to be sent to the server, and the subsequent response from the server. It can also describe a request generated by the browser, e.g. a timeout. The guard allows a transition to fire only when a mode has a particular value. For example, it can say that the user must be LoggedOn to fire this transition.

| **Page Reachability** | | |
|---|---|---|
| Category | CTL Formula | Sample Verification Rule |
| Reachability | EF (*PageVar* = *PageName*) | There is at least one navigation path from the start page to Queue page.<br>EF (Page = Queue) |
| Dead End | AG EF (*PageVar* = *pageName*) | Along any path from start point it is always possible to reach Queue page. (Queue page is reachable from any other page.)<br>AG EF (Page = Queue) |
| | AG ((*PageVar* = *Page1Name*) -> EF (*PageVar* = *Page2Name*)) | Along any path from start point, it is possible to reach Queue page after visiting Logoff.<br>AG ((Page = LogOff) -> EF (Page = Queue)) |
| Reach certain page in certain number of steps | AG ((*PageVar* = *Page1Name*) -> AX (*PageVar* = *Page2Name*)) | After Order Step 1 page the only possible next page is Order Step 2 page.<br>AG ((Page = OrderStep1) -> AX (Page = OrderStep2)) |
| | AG ((*PageVar* = *Page1Name*) -> EX (*PageVar* = *Page2Name*)) | It is possible to visit sign in page from sign off page.<br>AG ((Page = LogOff) -> EX (Page = LogOn)) |
| | AG ((*PageVar* = *Page1Name*) -> EF EX EX (*PageVar* = *Page2Name*)) | It is possible to visit Queue page within 2 links from Movie Home page.<br>AG ((Page = MovieHome) -> EF EX EX (Page = Queue)) |
| **Mode Reachability** | | |
| Reachability | EF (*ModeVar* = *ModeName*) | It is possible for a user to be in "logged-on" mode sometime.<br>EF (LogonStatus = LoggedOn) |
| Dead End | AG EF (*ModeVar* = *ModeName*) | It is always possible for a user to be in "logged-off" mode on all navigation paths.<br>AG EF (LogonStatus = LoggedOff) |
| | AG ((*ModeVar* = *Mode1Name*) -> EF (*ModeVar* = *Mode2Name*)) | It is possible for a user to switch to "logged-off" mode after the user is in "logged-on".<br>AG ((LogonStatus = LoggedOn) -> EF (LogonStatus = LoggedOff)) |
| **Page/Mode Coupling** | | |
| Allowed page and mode pairs | AG ((*PageVar* = *PageName*) -> (*ModeVar* = *ModeName*)) | A user must be logged-on to visit account page.<br>AG ((Page = Account) -> (LogonStatus = LoggedOn)) |
| Mode change only within certain pages | !E(!(PageVar = PageName) U ((ModeVar = ModeName) & !(PageVar = PageName))) | A user must visit Logon page to change to logged-on mode.<br>!E (!(Page = Logon) U ((LogonStatus = LoggedOn) & !(Page = Logon))) |
| **Page or Mode Sequences** | | |
| One page must be visited before another | !E (!(PageVar = Page1Name) U ((PageVar = Page2Name) & !(PageVar = Page1Name))) | A user must visit Signin page before visiting Queue page.<br>!E (!(Page = Logon) U ((Page = Queue) & !(Page = Logon))) |

Table 4: Examples on the properties checked by FARNav [HH06]

**Model optimization:** to scale the model the authors take benefit from the multilevel hierarchy of classical StateCharts to better manage large web applications, and they suggest a future work to deal with this problem by using hierarchal substates, or slicing on StateCharts, but they did not provide any further details.

**Tool support:** using existing model-checking tool (SMV).

**Features modelled:** Adaptive navigation; all features that can be checked for navigation but taking into account some events and conditions such as user state, examples on properties checked by FARNav are shown in Table 4.

11. **Notation**: control flow graph, data flow graph, and finite state machines, object state diagram

**Kung et al.** in [KLH00] propose a model that extends traditional test models, such as control flow graph, data flow graph, and finite state machines to web applications for capturing their test-related artifacts. Based on the proposed test model, test cases for validating web applications can be derived automatically. In this methodology, both static and dynamic test artifacts of a web application are extracted to create a Web Test Model (WTM) instance model. Through the instance model, structural and behavioral test cases can be derived systematically to benefit test processes. Test artifacts are represented in the WTM from three perspectives: the object, the behavior, and the structure.

- From the object perspective, entities of a web application are represented using object relation diagram (ORD) in terms of objects and inter-dependent relationships. In particular, an ORD = (V, L, E) is a directed graph, where V is a set of nodes representing the objects, L is a set of labels representing the relationship types, and ($E \subset VxVxL$) is a set of edges representing the relations between the objects, There are three types of objects in WTM: client pages, server pages, and components, to accommodate the new features of web applications, new relationship types are introduced in addition to those in the object-oriented programs. The new relationship types, navigation, request, response, and redirect are used to model the navigation, HTTP request/ response, and redirect relations introduced by web applications, respectively. Thus, in the ORD, the set of labels L = I, Ag, As, N, Req, Rs, Rd, where I: inheritance, Ag: Aggregation, As: association.

- From the behavior perspective, a page navigation diagram (PND) is used to depict the navigation behavior of a web application. The PND is a finite state machine (FSM). Each state of the FSM represents a client page. The transition between the states represents the hyperlink and is labeled by the URL of the hyperlink. The PND of a web application can be constructed from an ORD. To deal with the dynamic navigation (the construction of client pages can be dynamic at runtime based on the data submitted along with the HTTP requests or the internal states of the application. Hence, the same navigation hyperlink may lead to different client pages). To model this behavior a guard condition enclosed in brackets is imposed on the transition in the PND. The guard condition specifies the conditions of the submitted data or internal system states that must be true in order to fire the transition. To detect the errors related to navigation behavior a navigation test tree is employed. A navigation test tree is a spanning tree constructed from a PND, by analyzing the tree; they can check some properties, such as reachability and deadlock, of the navigation behavior. At the same time, a set of object state diagrams (OSDs) are used to describe the state behavior of interacting objects. The OSD is similar to StateChart. It can represent the state-dependent behavior

of an object in a web application. The state-dependent behavior for an aggregate object then can be modeled by a composite OSD (COSD) of the corresponding OSDs.

- The structure perspective of the WTM is to extract both control flow and data flow information of a Web application. To capture control flow and data flow information, the Block Branch Diagram (BBD) and Function Cluster Diagrams (FCD) are employed in the WTM. The BBD is similar to a control flow graph. It is constructed for each individual function of a Web application to describe the control and data flow information, including the internal control structure, variables used/defined, parameter list, and functions invoked, of a function. Therefore, the BBD can be used for traditional structural testing of each individual function; the FCD is a set of function clusters within an object. Each function cluster is a graph G = (V,E), where V is a set of nodes representing the individual functions and E $\subset$ V $\times$ V, is a set of edges representing the calling relations between the nodes.

**Features modelled:** reachability and deadlock, dynamic navigation, client and server script execution, session control properties, e.g. To check whether or not the inactive period of the current session is over a time limit, say, 5 minutes. If the inactive period is greater than the time limit, the HTTP request must be redirected to an authentication page to verify the user again for protecting the user's privacy.
$(\exists s \in Session)((s = this.Session() \wedge s.Inactive() > 5) \rightarrow this.redirect(Auth))$
The this.Session () is a function that returns a session object; s.Inactive() is a function that returns the inactive period for the session object s. The this.redirect (Auth) specifies that the HTTP request is redirected to the Auth server-page.

12. **Notation:** Communicating automaton FSM

**May Haydar et al.** in [HPS04] devise an algorithm to convert the observed behavior, which they called a browsing session, into an automata based model. In case of applications with frames and multiple windows that exhibit concurrent behavior, the browsing session is partitioned into local browsing sessions, each corresponding to the frame/window/frameset entities in the application under test. These local sessions are then converted into communicating automata. The constructed models can also be used for other purposes such as documenting, testing, and maintenance of web applications. They did an implementation for a framework which includes the following steps: The user defines some desired attributes through a graphical user interface prior to the analysis process. These attributes are used in formulating the properties to verify on the application. A monitoring tool intercepts HTTP requests and responses during the navigation of the Web Application Under Test (WAUT). The intercepted data are fed to an analysis tool, that continuously analyzes the data in real time (online

mode), incrementally builds an internal data structure of the automata model of the browsing session, and translates it into XML-Promela. The XML-Promela file is then imported into aSpin, an extension of the Spin model checker. ASpin then verifies the model against the properties, furthermore the model checking results include counter-examples that facilitate error tracking.

**Model optimization:** by partially analyzing the model (system of communicating automata).

**Tool support:** a framework that is composed of; GUI to collect desirable properties from the user, network monitoring tool to intercept HTTP request and response, analysis tool that builds the communicating automata based on the received data. The model is fed into aSpin for verification.

**Features modelled:** reachability properties; deadlocks and livelocks; frames behavior related properties, and their mixture; form filling, as well as various intricate and thus difficult to detect properties related to the concurrent behavior of the different entities of the WA. Properties can also be attribute-related where the user specifies some desired features based on page attributes.

13. **Notation: FSM**

**FSMWeb** [AOA05] builds hierarchies of Finite State Machines (FSMs) that model subsystems of the web applications. This approach which aims to testing web applications proceeds in two phases. Phase 1 builds a model of the web application. This is done in four steps: (1) the web application is partitioned into clusters, (2) logical web pages are defined, (3) FSMs are built for each cluster, and(4) an Application FSM is built to represent the entire web application. Phase 2 then generates tests from the model defined in Phase 1.

**Model optimization:** by using hierarchical collection of aggregated FSMs with constraints.

**Tool supprt:** FSMWeb prototype.

**Features modelled:** modeling and testing of static links (HTML/ HTML), dynamic links (HTML/software), dynamically created HTML (software / HTML), User/time specific GUIs (software + state / HTML), software connections.

14. **Notation: Directed graph**

**In Veriweb**[BFG02] Systematic Web-site exploration is performed under the control of VeriSoft, a previously existing tool for systematically exploring the state spaces of concurrent/reactive software systems. In VeriSoft the state space of the system is defined as a directed graph that represents the combined behavior of all the components of the system being tested. Paths in this graph correspond to sequences of operations (scenarios) that can be observed during executions of the system. In web site the state space is the set of web pages (statically or dynamically generated) in the site that can be reached from some initial page, so reachable pages are the states of the web-site state space, while the set of possible actions from a given page determined by ChoiceFinder defines the set of transitions from the corresponding state.

Whenever a new web page is reached and a new set of possible actions is determined; VeriSoft records this set of actions and executes one of them. The process is recursively repeated on the next page until some depth in the state space (i.e., number of successive actions) is reached. At that point, VeriSoft re-initializes the state of the web-site and starts executing a new scenario from that initial state. By repeating this process, all possible execution paths of a web application up to the given depth can eventually be exercised and checked.

Since the state space of a web site can be huge in practice, VeriWeb supports various techniques and heuristics to limit the size of the part of the state space being searched. Three main tasks are involved in exploring paths in a web site: searching (i.e., determining the set of possible actions and systematically go through these), execution (i.e., executing actions), and error handling (i.e., detecting and reporting errors), VeriWeb allows checking for many different types of errors, from errors in an isolated web page (e.g., the presence of a string pattern for an error, conformance to accessibility guidelines), to errors that involve a navigation path (e.g., constraints on length of the deepest paths in the site)

**Model optimization**: The level of pruning can be tuned by defining SmartProfiles, and profile policies (search constraints); restricting the search by not following URLs outside of a set of domains; eliminating links that match some pre-defined set of regular expressions; and setting a limit on the number of links to be followed in each page. A possible optimization is to record Visited URLs and prevent the search from exploring successors from the same URL more than once. In the current prototype, they rely on VeriSoft to limit the depth of the search and guarantee the termination of the search process.

**Features modelled:** the authors deal with pages that contain forms and client-side scripts; check three type of errors:

**navigation (execution) errors**, such as : page not found; unsuccessful form submission; and constraints on length of the deepest paths in the site.
**Page errors(application specific errors)**, such as "cannot connect to database"; "invalid customer"; constraints that must hold throughout the web site such as all pages must contain a navigation bar.
**Error logging**, is performed at three different levels: by VeriSoft (error traces);by the Navigator(Smart-Bookmarks); and by the web Proxy (Cache of pages retrieved).

15. **Notation**: Extended directed graph (Webgraph)

**Sciascio et al.** state that due to complexity of the hypertextual structure of the web, a web application cannot be modeled using a simple graph structure where nodes represent pages and arcs represent hyperlinks [SDMP02],[SDMP03]. Using frames, makes a window be composed by several pages. To solve this question the authors identified a new kind of object in a web application and consequently a new kind of state in the model, which is the "window" state. A generic window could be divided into one or more frames, where one or more web pages can be loaded. Sciascio et al., represent the system as a Kripke structure and model a web site as a Webgraph. Model checking is reformulated as checking that each initial state satisfies the specifications. They adopt Computation Tree Logic (CTL) as language to define the properties to be verified. The proposed formal method has been deployed in AnWeb, a tool for automatic support in the design of web applications. The tool provides an interface to the NuSMV model checker. The system parses the HTML source code of web pages, including code for dynamic pages, builds the model in NuSMV input language and provides the proper CTL specifications to the NuSMV tool.

**Tool support**: AnWeb

**Features modelled**: the authors provide examples on checking static properties such as broken links, reachability, frames Consistency, and for dynamic features they provide a model for the generation of dynamic pages (pages which redirect the navigation depending on conditions in the input form, and not those whose content is automatically generated by getting data from databases and processing user input.) However, they do not provide any examples on how to check the dynamic features.

16. **Notation**: FSM

**Sciascio et al.** in [SDM+05] and [CMRT06] try to answer another essential question in WA modelling, that is to distinguish between links connecting to other web pages, and links triggering an action of the server, for example, the download of a file or login operations). Hence, the authors extend the WA model to include "action" states representing actions performed in a specific web page, typically said "Server Page". In [SDM+05] , WAs are modeled as FSM where pages, links, windows and actions are states. The authors also proposed a mathematical model of a WA partitioning the usual Kripke structure into windows, links, pages and actions. Then they specify properties to be checked in a temporal logic, Computation Tree Logic (CTL). Verification is performed by adapting the NuSMV model checker to the proposed formalism. An implemented system embeds a parser to perform the automated parsing of the XMI output of the UML tool, and to automatically build the NuSMV model to be verified with respect to specifications.
In [CMRT06] **Castelluccia at al.** introduce a tool (WAVer) that is able to transform UML diagrams in XMI files, and to turn them into corresponding web application graphs (WAGs). Then the tool translates the WAG in a NuSMV model, which is finally used as input for the model checker NuSMV. It automatically performs verification of CTL specifications.

**Tool support**: WAVer.

**Features modelled**: in addition to those features supported by AnWeb, Sciascio et al. in [SDM+05]are able to check some properties of web application design such as:

- to check whether the access to private page occurs through a login; e.g. we must find some private information after a login action.

In WAVer [CMRT06]the model is extended to be able to check some properties related to access control such as:

- administrator can access to resources belongings to every authorized user.

17. **Notation**: UML meta model + CFG.

**Tonella and Ricca** [TR04] propose a two-layer-model for modelling and analyzing web application. The first layer "higher level", models the structure of a web application in terms of its composing pages and its navigation links. The first layer model is obtained according to the approach descried in [TR02], and that by extracting a UML instance model of web application based on a proposed UML-meta model in [TR02]; a static analysis for the HTML code that is dynamically generated by the server programs is performed. The Web application model produced in this way can be enriched with the transition probabilities, obtained from the statistical information dumped

by the Web server during execution. This approach has some weaknesses, there are some cases where the web server needs an additional tracing mechanism. These cases happened when caching is interposed between client and server, because this part of user navigation is not visible in the access log, and has to be reconstructed heuristically. Moreover, the input values passed to the server programs are not visible in the access log, if parameter passing is by POST instead of GET. In this approach, the input values used during model extraction are not being generated automatically. The source code running on the browser(e.g., Javascript and Applets),and that executed by the web server are not analyzed currently. Their analysis would allow treating these components as white-boxes.
The second layer "low level" model, is being obtained by considering the execution flow followed at the server and client side. In this model, nodes in the control flow model of a web application represent the statements that are executed either by the web server, or by the client (Internet browser). Edges represent control transfer. Since the execution on the server involves one (or more) server side languages (e.g., PHP and SQL), and the execution on the client involves additional languages (such as HTML and JavaScript), the control flow model has different node kinds ("colors"), according to the programming language of the respective statements. In addition to the sequential; deterministic control transfer, possibly controlled by conditional or loop statements; some statements in web application determine the "registration" of functions or scripts/pages with respect to given graphical events.In order to perform the required analysis and testing, two prototyping tools are implemented ReWeb and TestWeb. TestWeb [RT01b] consists of two modules: the Test generator and the Test Executor. The Test generator is able to generate test cases from the UML model of a Web application. The user has to add some information to the UML model produced by ReWeb to prepare it for testing purposes. The user also has to choose a test criterion. The Test generator computes the path expression of the model and uses it to generate sequences of test cases which satisfy the coverage criteria chosen by the user. Input values in each URL sequence are left empty by the Test generator, and the user has to fill them in; those values include: specification of the page type when the distinction between static and dynamic pages cannot be obtained automatically, variables for each dynamic page whose content depends on some input value and attached conditions to the edges whose existence depends on the input values.

**Tool support:** ReWeb, TestWeb.

**Features modelled:**in addition to those properties checked in ReWeb, Client side execution and Server side execution are modeled.

18. **Notation:** UML meta model + UML state diagram.

    **Bellettini et al.** in [BMT04] propose a web application reverse-engineering tool WebUml, which is used to extract UML models (in particular class and state diagrams) with minimal user interactions. The meta-model that is being used to represent web application class diagram is similar to the Conallen model and like the Tonella and Ricca model. Class diagrams are used to describe the structure and the components of web application (e.g., forms, frames, Java applets, input fields, cookies, scripts, etc.), while state diagrams are used to represent the behavior and the navigational structures (client-server pages, navigation links, frames sets, inputs, scripting code flow control, etc. ). To build state diagram; the first operation in such construction consists of class diagram analysis to define application components that have fundamental states for web application evolution. The next step examines components to search for the events that change the state of the component itself, such as inputs inserted values, function scripting call, mouse clicks, etc... WebUml uses a mix of techniques based on source code static and dynamic analysis. Static analysis is performed through simple parsers based on a pattern matching scanner. Dynamic analysis is performed through source code mutational techniques combined with simulated web application execution. WebUml analyzes client side pages by elaborating the information extracted in the class diagram constructor modules, and if necessary (e.g., for active Web pages) applies specific static code analysis in order to define details about the client-side dynamic characteristics (scripting code, dynamic links, etc.). Server pages analysis is a Server pages analysis is a combination of: source code mutation, application executions WebUml dialogue with Web server) and traditional source code analysis. This technique avoids heavy language analysis, but needs the implementation of a simple map of mutant operators. The generated UML models may seem crowded, but they are also very rich in information.

    **Tool support:**WebUML.

    **Features modelled:**modeling of client-server pages, navigation links, frames sets, form inputs, scripting code flow control. This model is intended to be used in a testing tool that is under construction (TestUML).

19. **Notation**:UML-based Web Engineering(UWE).

    **Knapp and Zhang** in [KZ06] propose a systematic method of merging the separate models of web application into one integrating model by using graph transformation rules. The transformation is done from the context of UWE meta-model (navigational model and business model) into a UML state machine. First they transform the navigation model into a basic state machine, and then they integrate the business processes into this state machine as submachine states. The final model includes the static nav-

igation structure as well as the dynamic behavior of a web application model. The authors choose the final model to be in the UML state machine because there are tools that do the conversion from this model into input models for the model checkers SPIN and –for real time state machine – UPPAAL.

This method uses UML-based Web engineering methodology (UWE), which is a design methodology. The features that this method is able to capture are the same as those captured by UWE; as well as those features that appear as a result of the integration process proposed by this method. UWE is able to model the static navigation of the web application by its navigational model; where the behavioral modeling is only mentioned with respect to defining the sequence of navigation by mean of constraints. The only dynamic features are currently modeled by the dynamic presentation model and that by employing UML StateCharts for describing the activation of navigation. The authors present three examples for using the model checker over their final model; they check for page reachablity and for session control properties.

**Tool support:** ArgoUWE, Hugo/RT(translates state machines, collaborations, and assertions into input models for the model checkers SPIN), Spin.

**Features modelled:** page reachablity and session control properties over the integrated model.

20. **Notation:** Regular Expressions.

**Ye Wu and Je Outt** in [WO02] are interested in modeling the dynamic aspects of web application and specifically the dynamic interaction among clients and servers, and dynamically generated client components. Thus this method assumes that clients and servers interact through HTML pages. The technique is based on identifying atomic elements, which are defined as a static HTML file or a section of a server program that prints HTML and has an "all-or-nothing property", that is, either the entire section is sent to clients or none of the section is sent. An atomic element may be a constant HTML section, or it may be an HTML section that has a static structure but may contain content variables. A content variable is a program variable that provides data to the HTML page but not structure. These elements are dynamically combined to create composite web pages using sequence, selection, aggregation, and regular expressions. For a server program consisting of four atomic elements this method computes all possible complete HTML files that can be generated by the component using the following composition rule : $P \rightarrow p1.(p2 \mid p3)^*.p4$.
The actual dynamic composition is affected by the control flow of the server component.The authors also provide a definition for the dynamic interaction, which can be represented by a set of transition rules.

if p and q are composite sections and s is a servlet or other software these transitions can be:

- Link Transition $p \Rightarrow q$: Invoking a link in p causes a transition to q from the server to the client. If p can invoke one of several static or dynamic pages, q1, q2, . . . , qk, then the link transition is represented as $p \rightarrow q1 \mid q2 \mid ... \mid qk$. Link transition can be an HTML link defined via the $< A >$ tag, or an action link defined in a $< FORM >$ section.
- Composite Transition $s \rightarrow p$: The execution of s causes p to be produced and returned to the client. The servlet s will normally be able to produce several composite web pages, which can be represented as $s \rightarrow p1 \mid p2 \mid ... \mid pk$.
- Operational Transition $p \rightharpoonup q$: The user can inject new transitions out of the software's control by pressing the back button or the refresh button.

Based on the above definition, a web application W is modeled as a triple {S, C, T}, where S is the start page, C is a set of composition rules for each server component, and T is a set of transition rules. An example of such approach is described in Figure 4, in this example a small application includes query servlet(A), and another servlet that processes the email to the instructor, which is not shown in the figure; HTML file (B); the approach model is shown in (C).

The testing relies on the descriptions of the web software behavior to define tests as sequences of user interactions that begins with the start page, and uses composition and transition rules to reach the desired page, this sequence is called derivation. For example, there are several derivations for the case when a student enters an incorrect ID password pair:

$S \Rightarrow GradeServlet \rightarrow p1.p3.p4 \Rightarrow S$
$S \Rightarrow GradeServlet \rightarrow p1.p3.p4 \Rightarrow SendMail...$
$S \Rightarrow GradeServlet \rightarrow p1.p3.p4 \rightharpoonup PreviousS \Rightarrow$
$GradeServlet \rightarrow p1.p2.p4 \Rightarrow S$

A major advantage of this model is that, it relies on the principles of HTTP and HTML construction, thus it is independent of software implementation technology. The model includes only Java servlet components and doesn't deal with other technologies such as XML, JSP,ASP. The model was also built on the assumption that webapplications are based on HTTP, which is a stateless protocol, so it did nothing to deal with session control mechanisms such as cookies.

**Features modelled:** modeling and testing for static navigation properties, dynamic navigation properites, user operations, server and client side execution.

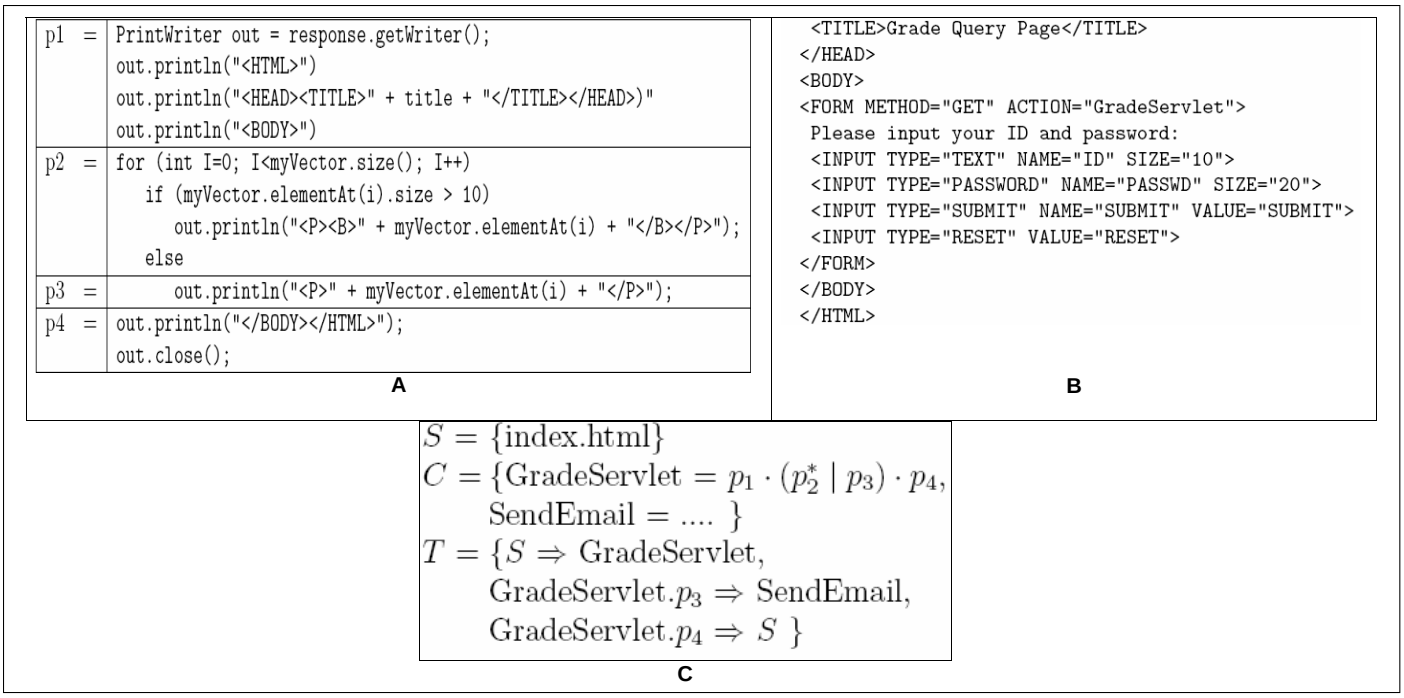


Figure 4: Example on **Ye Wu and Je Outt** method [WO02]

21. **Notation:SDL**.

**Syriani and Mansour** in [SM03] use the SDL model of a web application to represent all the web pages of the system and their relationships with each other. Each web page is represented by an agent, and the hyperlinks between pages are represented by signals. A hyperlink in a web application represents a navigational path through the system. This relationship is represented in the SDL model with a signal sent through a channel association. The signals may contain parameters. For example, such parameters may contain user name and passwords that are sent within the signals to login to a server. This link association originates from a client page and points to either a client or a server page. The procedure embodied in both the client and the server can be modelled using composite states. This is done by modelling the transition from one state to another in order to control the internal procedures of both the client and the server. These steps may contain input signals, output signals, procedure calls, etc. A Message Sequence Chart can be used to model the sequence steps of moving between agents or states. If a client is accessing a server, a communication signal is fired. Upon receiving this signal, the server should reply back with another signal. The contents of this signal can also be specified. This model helps in verifying the consistency of a web application implementation with it's specification. The authors include only one example of how to use the existing Testing and Test Control Notation (TTCN) of the SDL to verify that the implementation of the web application conforms to it's specification, and that by generating test cases. For example, if there is a login request

which is controlled by time - so that a time interval is added to the representation. They check if the client receives a response of either login-confirm or login-error that satisfies the specification of the web application and within the time interval, then the test passes; else if time out happened the test fails; if other responses are received, nothing happened until time out

**Tool support:** use techniques and tools that support SDL.

**Features modelled:** the approach models pages; hyperlinks; static and dynamic navegation.

## 5.2 Comparison Results

In this work we present two comparative studies. Table 5 shows the first one, where 21 methods are categorized according to the level of web application modelling into interaction behavior modelling methods; navigation modelling methods; content modelling methods and hybrid modelling methods( methods that models more than one level). In each category methods are sorted according to the notation used by each method. At the same time a comparison between these methods is done based on other criteria such as: application for the method either for testing or for verification; whether the source code is required for the analysis or not; the way the method solves the state space explosion problem; and finally, according to whether there is a tool that supports these methods or not.

The second comparison, shown in Table 6, aims at a comparison of the more specific details between methods in the same category in particular, and with other methods in other categories in general. The comparison is based on a combination of feature type and the level of web application modeling, using the comparison criteria outlined in Section 3 as desirable properties for web site modeling.

In the remainder of this section we discuss and compare the characteristics of the methods summarized in these tables. Our presentation is organized by the levels in Table 5, that is, we first discuss interaction modelling methods, then navigation modelling methods, followed by content modelling methods and finally hybrid methods. The categories are not disjoint; some methods are discussed more than once since they have aspects that address multiple levels.

### 5.2.1 Interaction behavior Modelling Methods

Dealing with user operations (interactions) is very important. Such interactions are problematic, for example: clicking the back button forces the computation to resume at a prior interaction point; submitting multiple clones then clicking the back button causes computations at the same interaction point to resume many times. These operations happen on the browser and are not reported to the web application. Consequently, the web application will be affected in an unexpected manner, so any modelling methods that do not take

| Method Name | Feature type | Notation | Level | Application | Source code required | Model optimization | Tool support |
|---|---|---|---|---|---|---|---|
| [GFKF03] | Interaction | Abstract model, use lambda calculus | Interaction Behavior | WA interaction with the browser | No | No | Prototype |
| [LK04] | Interaction | WebCFG | Interaction Behavior | Verification | Yes | Yes | Implement a model checker |
| [CZ04] | Interaction +Static | Labeled transition | Interaction + static (Navigations) | Testing and verification | No | Yes | None |
| [BA05] | Interaction | UML(WS structure) OCL( behavior of the model) | Interaction Behavior | Verification for user interaction( Amazon + Orbitz bug) | No | Yes | UML2Alloy |
| [ABF05] | Static | Partial rewriting | Content | WS verification Tool(GVerdi) | Yes | No | GVerdi |
| [Con99 ] | Static | Extended UML | Structure (Navigation) | Analysis | No | No | Rational Rose Tools |
| [BMT04] | Static + dynamic | UML-meta Model + UML state diagram | Structure (Navigation) | Analysis & Testing | Yes | No | WebUML |
| [RT00] | Static | Directed graph | Structure (Navigation) | Analysis + can be use for verification & testing | No | No | ReWeb |
| [dA01] and [dAHM01] | Static | Directed graph With Webnodes | Structure (Navigation) | Verification | No | No | MCWeb |
| [SDMP02] | Static + dynamic | Web graph | Structure (Navigation) | WA design Verification | No | No | AnWeb |
| [SDM+05] and [CMRT06] | Static + dynamic | (WAG)WA graph + extension to Kripke structure | Structure (Navigation) | WA design Verification | No | No | WAVer + SMV tools |
| [WP03] | Static + dynamic | Extended StateCharts | Structure (Navigation) | Design Verification | Yes | Yes | SWCEditor |
| [HH06] FARNav | Static + dynamic | StateC harts | Adaptive Navigation | design and implementation Verification + testing | No | Yes | Existing SVM model-checking tools |
| [SM03] | Static + dynamic | SDL | Structure (Navigation) | Testing and verification | Yes | No | Existing SDL Support tool |
| [KLH00] WTM | Static + dynamic | Control flow graph, data flow graph, and finite state machines OSD( object state diagram) | Static and dynamic Behavior, Dynamic Navigation | Testing | Yes | No | None |
| [BFG02] Veriweb | static + dynamic | Directed graph | Navigation + Behavior | WS testing | Yes | Yes | VeriSoft + web Navigator + ChoiceFinder + SmartProfiles |
| [HPS04] | Static+ dynamic | System of communicating automata | Navigation + Behavior | WA Verification | No | Yes | Fame Work with GUI + network monitoring tool + analysis tool |
| [AOA05] FSMWeb | static + dynamic | hierarchies of Finite State Machines (FSM) | Navigation + Behavior | System level testing | No | Yes | Prototype |
| [WO02] | Interaction + static + dynamic | Regular expression | Interaction + dynamic Behavior | Can be used for testing + implementation + impact analysis | Yes | No | None |
| [TR04] And [TR02] | Static + dynamic | (model navigation layer) + CFG (client & server code) | Structure (Navigation)+ Behavior | Testing | Yes | No | ReWeb + TestWeb |
| [KZ06] | Static + dynamic | Extended UML (UWE) | Structure (Navigation) + Behavior | Design Validation and Verification | No | No | ArgoUWE + Spin or UPPAAL |

Table 5: Survey Result (categorized according to modelling level)

| Method Name | Desirable Features of Web application Modeling | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Static Navigation Properties* | | | | | *Dynamic Navigation* | | *Interaction Navigation* | *Static content prop.* | | *Dynamic content properties* | | | | *Instructions processing* | | *Security properties* | | |
| | *Broken links* | *Reachablity* | *Frames consistency* | *Form filling* | *Longest path* | *System input* | *User input* | *HTML + user operations* | *Incomplete WP* | *Incorrect WP* | *Incomplete WP* | *Incorrect WP* | *New connection* | *New content* | *Server-side execution* | *Client-side execution* | *Access control* | *Session/cookies* | |
| **[GFKF03]** | | | | Y | | | | Y | | | | | | | | | | | |
| **[LK04]** | | | | | Y | | | Y | | | | | | | | | | | |
| **[CZ04]** | Y | Y | Y | | | | | Y | | | | | | | | | | Y | |
| **[BA05]** | | | | | | | | Y | | | | | | | | | | | |
| **[ABF05]** | | | | | | | | | Y | Y | | | | | | | | | |
| **[Con99 ]** | | | | | | | | | | | | | | | | | | | |
| **[BMT04]** | Y | Y | Y | Y | | | Y | | | | | | | | Y | Y | | | |
| **[RT00]** | Y | Y | Y | | | | | | | | | | | | | | | | |
| **[Da01] and [dAHM01]** | Y | Y | Y | | Y | | | | | | | | | | | | | | |
| **[SDMP02]** | Y | Y | Y | | Y | | Y | | | | | | | | | | | | |
| **[SDM+05]** | Y | Y | Y | | Y | Y | Y | | | | | | | | | | | | |
| **[CMRT06]** | Y | Y | Y | | Y | Y | Y | | | | | | | | | | Y | | |
| **[WP03]** | Y | Y | Y | | | Y | Y | | | | | | Y | Y | | | | | |
| **[HH06] FARNav** | Y | Y | Y | | | Y | Y | | | | | | | | | | | | |
| **[SM03]** | Y | Y | Y | | | | Y | | | | | | | | | | | | |
| **[KLH00] WTM** | Y | Y | Y | | | Y | Y | | | | | | | | Y | Y | | Y | |
| **[BFG02] Veriweb** | Y | Y | Y | Y | Y | | Y | | Y | | | | | | | Y | | | |
| **[HPS04]** | Y | Y | Y | Y | | | Y | | | | | | | | | Y | | | |
| **[AOA05] FSMWeb`** | Y | Y | Y | Y | | Y | Y | | | | | | Y | Y | Y | Y | | | |
| **[WO02]** | Y | Y | Y | | | Y | Y | Y | | | | | | Y | Y | Y | | | |
| **[TR04] And[ [TR02]** | Y | Y | Y | | | | | | | | | | | | Y | Y | | | |
| **[KZ06]** | Y | Y | | | | | | | | | | | | | | | | Y | |

Table 6: Comparison Result

into account this kind of behavior are considered incomplete and unrealistic. Researchers are aware of this kind of problem and try to include user operations in their proposed web application models.

Graunke et al. in [GFKF03] try to detect data inconsistency problems, such as the Orbitz bug. They detect these problems dynamically through changes to the server run-time system. They propose an abstract model where they encode user operations using rewriting rules; in addition their method is able to detect other inconsistency problems that are related to form filling. Licata and Krishnamurthi, in [LK04], built a model checker which benefits from Graunke et al. model in order to reduce user operations into two main rules: submit and switch. Their method is different from Graunke et al. in that it is a static method and can provide guarantees about all possible execution sequences by building a CFG to model the web application. User operations are added to the CFG, extending it to what is called WebGFGA.

Bordbar and Anastasakis in [BA05] create an abstract model (ADI) to depict the interactions between the browser and the business logic. This model consists of four classes: the browser with its functionality; the business logic that relates to the browser and its data content; the data that are exchanged between the server and the browser; and the generic functionality of the web page that contains data which could be altered from the user interface. While Licata and Krishnamurthi in [LK04] build their own model checker, Bordbar and Anastasakis use Alloy analyzer in order to find the interaction bugs. The main difficulty of this method is the process of building the ADI model from the Platform Independent Models (PIM) of web applications which are large and complex. The construction process needs the projection of the PIM and the deletion of the unrelated model elements, which is currently done manually.

Chen and Zhao in [CZ04] model the user's interactions with web browsers using a much more complete model; as well as modelling the back button, forward button and URL rewriting functionalities, the method is different from other methods in its ability to represent history stack and its impact on the navigation, the local cache and its influence on the freshness of the web pages, and the authentication sessions. While this method builds a navigational model taking into account the interaction with the browser, dynamic links are not being taken into account in the assumed page navigation diagram, and in the functionality provided by session/cookie techniques, the authors have chosen only the session control.

### 5.2.2 Navigation Modelling Methods

Conallen [Con99] proposes an extension of the UML notation to represent web application components with both dynamic and static features. He does not present a modelling method for any of the web application development phases. On the other hand, his extension was

the basis of many modelling methods that are applied in different phases of web development. The main benefit of the method is the feature which allows representation of all the components of a web application using a standard UML notation.

Like Conallen, Tonella and Ricca [TR02] proposed a UML- Meta model to model web application, and specifically for static navigation. The main difference between the two approaches is that Conallen's model aims at describing the web application from the design point of view, where he didn't propose a method for the design nor for the navigation of a web application. While Tonella and Ricca use their model in a reverse engineering method, in order to extract a model for the web application for the aim of maintenance and evolution. So their aim by their model is the analysis rather than the design, and specifically on modeling and analyzing the navigation features.

The Tonella and Ricca Method is semiautomatic; it needs a lot of user interaction to complete the process of model extraction, there are some cases where the web server needs an additional tracing mechanisms; these cases occur when caching is interposed between client and server and has to be reconstructed heuristically. Moreover, the input values passed to the server programs are not visible in the access log, if parameter passing is done by POST instead of GET. In this approach, the input values used during model extraction are not being generated automatically, though they need extensive user interactions. The source code running on the browser (e.g., Javascript and Applets), and that executed by the web server are not analyzed currently. Their analysis would allow treating these components as white-boxes.

Bellettini et al. [BMT04] use a model similar to Canollen and like the Tonella and Ricca model to extract an instance of this model from the analyzed web application, but it's different from the Tonella and Ricca method in that WebUML needs a minimal user interaction.

Castelluccia et al. [CMRT06] and Sciascio et al. [SDM+05] use the Canollen model in order to build a diagram for the web application, where the aim of this work is to do verification for the design of WA. In order to apply model checking techniques on any model, it has to be formal, so the authors implement in this work a component which did the conversion between UML diagram in XMI format into WAG, where WAG is translated into SMV model which is given as input to the NuSMV model checker, this component is called XMI2SMV.

A similar conversion idea was applied by Bordbar and Anastasakis in [BA05] which was described previously in the interaction modeling methods. They propose a tool UML2Alloy, but in this case the conversion is being done from a UML diagram into the Alloy model, which can then be model checked.

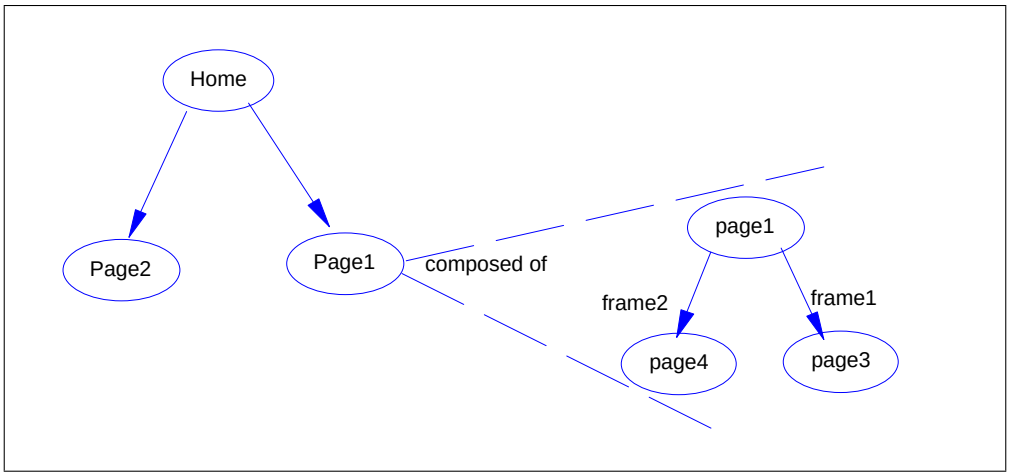


Figure 5: MCWEB Web Site Modeling [dAHM01]

UML diagrams provide a valid support to verify WA requirements; however, they need to be turned into a formal model; so other researchers prefer to start with a formal model rather than doing the conversion. In MCWEB [dAHM01] the authors model the web application using a web graph, instead of the simple, directed graph model (flat model) with web pages as nodes and links due to an anchor and frame (sub-frame) tags as edges. Their model supports natural connectivity analysis of the web, where webnode takes into account the hierarchical frame structure of the web page. Based on this model de Alfaro verifies desired properties expressed in constructive $\mu - calculus$ against static web applications. MCWEB tool downloads a web site from a given URL and builds an abstract representation of it in the form of a graph, figure 5 shows an example of the structure of a web site in this model.

Like de Alfaro, Ricca and Tonella [RT00] try to solve the issue of hierarchical frame structure of the web page but in a different way. In their model nodes and edges are split into different subsets. Nodes subsets are a set of all web pages; a set of frames for one web page; and a set of all frames. Edges are also split into three subsets according to the kind of target node; a set of hyperlinks between pages or a relation showing the composition of web page into frames(E1); a set of the relations between frames and pages; as they show which page in which frame is loaded(E2); and a set of relations showing the loading of a page into a particular frame(E3). The name of the frame is given as a label next to the link. This model is implemented in ReWeb, This tool can download and analyze a web site, and also provides graphical user interface for searching and navigating in the results. It's mainly proposed for the purpose of understanding web application, but later on, it's used to generate a UML model that is fed into TestWeb, a tool proposed by Ricca and Tonella [RT01b] for the purpose of web application testing. ReWeb is applied on static web pages with or without frame structure.

Figure 6 depicts an example of web site structure. The links between page3 and page5, and

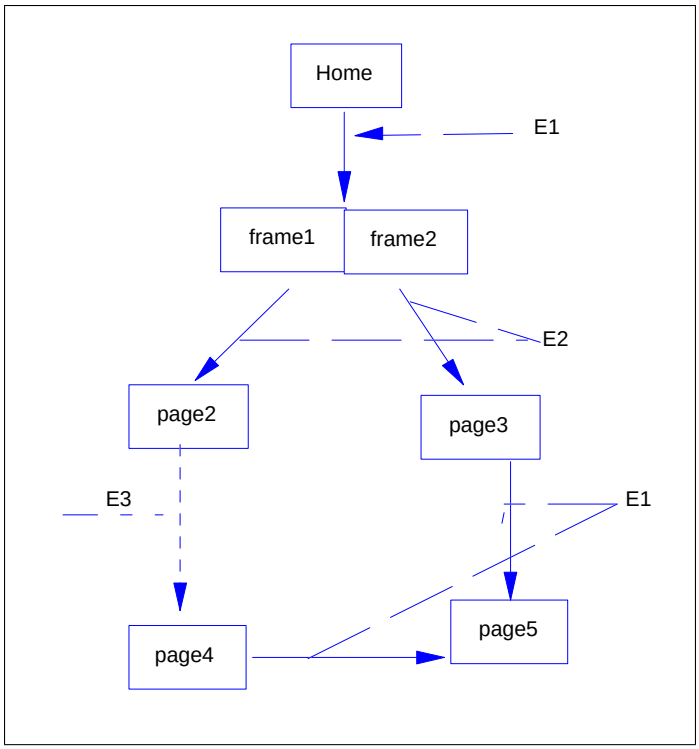


Figure 6: ReWeb Web Site Modeling in [RT00]

between page4 and page5 are normal navigation connections between HTML pages (El).The link between Home and frame1, frame2 represents the internal organization of page1 into the two frames frame1 and frame2 (El). The links between frame1 and page2 and between frame2 and page3 indicate that the pages initially loaded into frame1 and frame2 are respectively page2 and page3 (E2). Finally, the dashed edge connecting page2 to page4 and labelled with frame2 (E3) is used to show that a link in page2 does not result in the navigation within frame1 toward a different page, but rather produces the loading of page4 into frame2, with no regard to the page currently loaded into frame2.

Sciascio et al. in [SDMP02],[SDMP03]solve the same issue by proposing a new state - window-to correspond to a page that could be divided into one or more frames where one or more web pages can be loaded, so each node can be window, page or link, as in figure 7. In this work scripts are modelled, where client side scripts are modelled as static pages. For server-side scripts they consider modelling the dynamism of redirection action depending on user input (form). Other dynamic features that require white-box analysis for the scripts, such as server contact with the database and other resources, are not considered in this work; such pages are considered static pages.

Where in [SDM$^+$05] Sciascio et al. extend the previous model by adding actions to the set of states. So web applications are modelled as FSM where pages, links, windows and

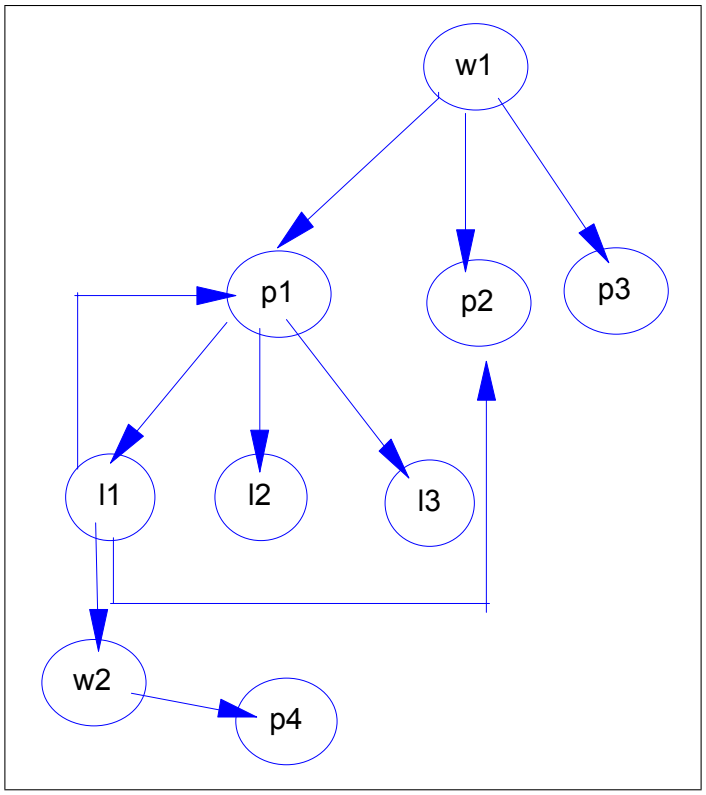


Figure 7: WAG Modeling [SDMP03]

actions are states as in figure 8.

WAG is extended in WAver tool [CMRT06]. Castelluccia et al. added some important features related to WA access policies. The extension was made by assigning some resources to two categories of users:

- Authorized users: they can view specific areas of the WA not accessible to anonymous users;
- Administrators: they can insert or cancel a new user, view the list of authorized users and access all the resources of the WA.

The WAG example becomes as in figure 9. after the model extension. By introducing this extension the Sciascio et al. model is able to represent an important feature related to access control, and is able to verify properties related to this feature using axioms formulated by CTL. The Sciascio et al method's main advantage is its ability to perform "a priori" verification of web application design by applying the verification process to the UML-design of web application in a single automated process using a verification tool WAver.

In general, graph-based models can be used to verify page reachability, dominators of the

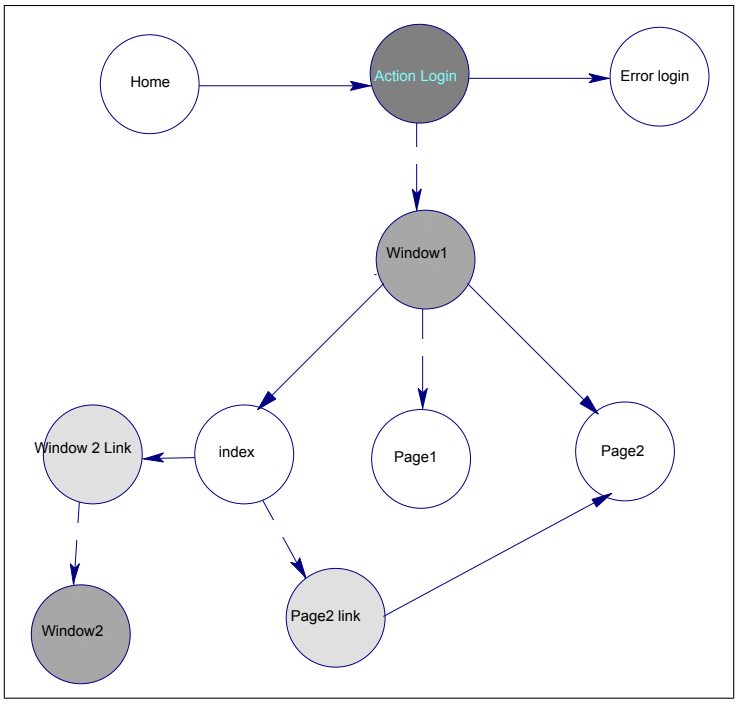


Figure 8: WAG Modeling [SDM+05]

navigation path, navigation path length, strongly connected components, broken links and frame errors. Also it is possible to do pattern matching to find out if the navigation model contains a diamond structure, tree structure or index structure.

Winckler and Palanque in [WP03] create an extension of StateCharts into what they call StateWebCharts, so their work is similar to Conallen's in that it creates an extension to an existing notation StateCharts instead of UML in the Conallen case; thus, they can help designers in building a formal model of web application that can be directly model-checked where the UML model can't. Currently SWC is used to describe the navigation between documents rather than the interaction between objects. They also create a tool - SWCEditor- that supports their proposed notation and helps designers create, edit, visualize and simulate SWC models.
Han and Hofmeister [HH06] also use a formal model for the navigation of web application. Their method (FARNav) uses StateCharts to model the adaptive navigation (web applications that can semi-automatically improve their organization and presentation by learning from visitor access patterns). They create their model by observing the behavior of the web application; and treating screens that provide a similar type of content as one web page. They try to scale their model by making use of the hierarchical feature of StateCharts. Like Sciascio et al. their model is converted into SMV model language CTL to be verified according to some axioms. They didn't propose an approach for the conversion into the SMV model like Sciascio et al.; they used an existing approach which has translation rules

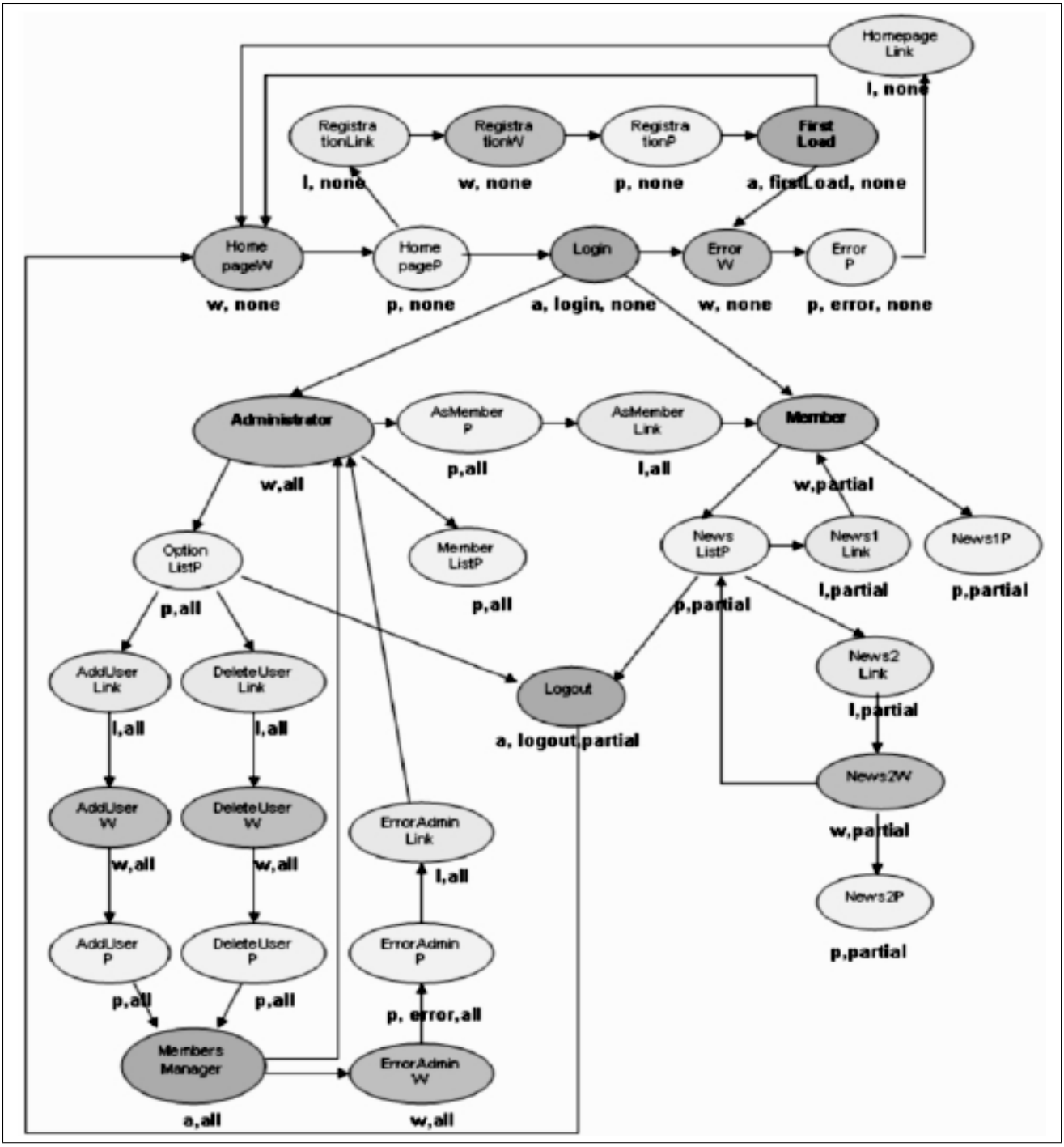


Figure 9: WAG Modeling [CMRT06]

of each element of a StateChart model. As FARNav uses StateCharts, the limitations of state machines' modelling capabilities make it difficult to verify certain properties that are easy to verify with a graph-based model such as de Alfaro , Sciascio et al, and Ricca and Tonella methods. For example, it is difficult to count the length of the navigation path with this approach. In addition none of the above models support adaptive navigation, or the automatic verification of adaptive navigation.

Syriani and Mansour in [SM03] use SDL to model the web application system; they are able to model pages, hyperlinks, behavior of the web page on the client side and the server side, client-server and distributed-server communication. They rely on SDL tools to do the testing for their model and to help them in verifying the consistency of a web application implementation with it's specification.

### 5.2.3 Content Modelling Methods

Few works address the semantic verification of web sites which is related to checking the completeness and correctness properties of the content of the web application. The only method that we present in our paper which tackles this issue is proposed by Alpuente et al. [ABF05] this work performs such checking by proposing a rewriting-based method to check only static web sites; we can recognize from Table 2 that there is a need for methods that perform such kinds of verification for both static and dynamic web content.

### 5.2.4 Hybrid Modelling Methods

Some researchers try to model the web application as a whole, taking into account all the modelling levels of this application, and try to solve the problem of state space explosion in some way; such as the model in FSMWeb [AOA05]. In this method, rather than building a flat model, it uses the idea of clustering related web pages into a logical web page, and then it builds hierarchies of finite state machines for the resulting logical web pages. The FSMWeb model is able to capture many static and dynamic features, but is still not able to cope with all the required features such as user operations (interactions), and security issue properties. An effort is needed to generate the logical web pages automatically as it is generated in the current method manually.

Another effort is proposed by May Haydar et al. [HPS04], where a system of communicating automata is generated to model local browsing sessions. These represent partitions of browsing sessions that are captured at run time from the observed behavior of static pages, dynamic pages with form filling (with GET and POST methods), frames and frameset behavior, multiple windows, and their concurrent behavior. This work also lacks the completeness property, as a consequence of not capturing other dynamic features, user operations, and security issue properties.

In VeriWeb [BFG02], Benedikt et al. present a dynamic navigation testing tool for web

applications. The main difference between VeriWeb and FSMWeb is that VeriWeb's testing is based on graphs where nodes are web pages and edges are explicit html links, and the size of the graphs is controlled by a pruning process. In contrast FSMWeb is based on FSM and uses hierarchies of FSM to reduce the state space size. VeriWeb is able to deal with static pages, forms and client-side scripts.

Ye Wu and Je Outt, in [WO02], present a modelling technique for web applications based on regular expressions. They model the behavior of web applications, consisting of merely dynamically generated pages for the purpose of functional testing. This work is different from FSMWeb, Haydar et al, and VeriWeb in its ability to deal with user operations, and its need of the source code for the analysis.

On the other hand, some works try to model web application from more than one level by using separate models. Tonella and Ricca propose a 2 layer-model [TR04]; the first layer is a UML model of web application for high level abstraction. This layer model is based entirely on static html links and does not incorporate any dynamic aspects of the software. The second layer is represented by multicolor CFG and is obtained by white-box analysis supported with information extracted from the access log of the server while the application is under execution. This work is different from FSMWeb and May Haydar et al. in that it performs white-box analysis and uses multiple models. This work is different also from May Haydar et al. in not being able to model the concurrent behavior of frames and multiple windows

In WTM [KLH00] Kung et al. developed their method based on multiple models of the applications under test. The models include Object Relation Diagrams, Object State Diagrams, a Script Cluster Diagram, and a Page Navigation Diagram. FSMWeb and Haydar et al methods are different from this work in that those methods do not require the source code to be available; their models are built depending on logical web pages rather than physical web pages, and they use the enhanced single FSM model instead of multiple models. Regarding implemented features, Kung et al. is different from FSMWeb in not being able to deal with dynamically generated web pages, and from Haydar et al. in not being able to deal with the concurrent behavior of multiple windows.

Even so WTM, Tonella and Ricca, Ye Wu and Je Outt in methods use a white-box approach in the analysis of web application, the navigational model obtained by Tonella and Ricca is static, whereas in WTM and Ye Wu and Je Outt the model is dynamic.

While these methods –WTM, Tonella and Ricca– try to model web applications using more than one model, the integration of those models and the validation of their interaction are not described clearly. In contrast, Knapp and Zhang [KZ06] propose a systematic approach to integrate a model for web application from separate models. They do this by using graph transformation rules on the UML-based Web Engineering meta-model to generate UML

state machine, which includes the static navigation in addition the dynamic behavior. The final model can then be validated formally, though this model still lacks checking of many dynamic properties, security issue properties, and interactions properties.

We are looking for a model that is able to capture all the desirable features of web application at all modelling levels, as well as being able to validate such a model using model checking. To the best of our knowledge such a model does not exist, but may be obtained by integrating some of the existing modelling techniques. In addition, web applications have the property of low observability, and that is due to the difficulty of tracking some outputs. Usually the output that is sent back to the user as html documents is being analyzed, but there is also another kind of output such as changing state on the server or in the database, and sending messages to other web applications and services; until now there is no research which address this issue.

# 6 Conclusions and Open Problems

Little work has been done to compare different modelling methods in the field of web development. To the best of our knowledge this is the first study which focuses on the process of giving a comprehensive review and a comparative study of modelling methods that are currently applied in the field of verification and testing, while all previous works was focusing on the development process in general, and on the design phase in particular.

Comprehensive reviews and comparative studies such as ours can help in highlighting the areas that need further research, and may help new researchers who are interested in the area to quickly get an idea of what has been done, and what could be done. This is especially so if the study is able to provide them with the strong and the weak points for each method, which may give them ideas on how to combine the strong points in a unified improved new modelling method.

We have surveyed 21 modelling methods that are currently applied in the field of verification and testing. A categorization and a comparative study were made according to two sets of criteria, and summed up into two tables. We found that this field is still in its infancy, while there is much that has been done in this area of research, up until now there is no complete modelling method that is able to capture all the desirable properties of web applications at all modelling levels. An integration of different modelling methods may be required in order to generate a new model that could be verified using model checking.

There is also a need for work on security modelling techniques that are able to deal with the complex, distributed structure of web applications, taking into account the concurrent access to web servers and the other resources that are attached to them.

# References


[ABF05] María Alpuente, Demis Ballis, and Moreno Falaschi. A rewriting-based framework for web sites verification. *Electr. Notes Theor. Comput. Sci*, 124(1):41–61, 2005.

[AOA05] Anneliese Amschler Andrews, Jeff Offutt, and Roger T. Alexander. Testing web applications by modeling with fsms. *Software and System Modeling*, 4(3):326–345, 2005.

[BA05] Behzad Bordbar and Kyriakos Anastasakis. MDA and analysis of web applications. In Dirk Draheim and Gerald Weber, editors, *Proceedings of the Trends in Enterprise Application Architecture*, volume 3888 of *Lecture Notes in Computer Science*, pages 44–55. Springer, 2005.

[BFG02] Michael Benedikt, Juliana Freire, and Patrice Godefroid. Veriweb: Automatically testing dynamic web sites. In *Proceedings of the 11th International World Wide Web Conference, Hawai, U.S.A.*, May 2002.

[BMT04] Carlo Bellettini, Alessandro Marchetto, and Andrea Trentini. Webuml: reverse engineering of web applications. In *SAC*, pages 1662–1669, 2004.

[CMRT06] Daniela Castelluccia, Marina Mongiello, Michele Ruta, and Rodolfo Totaro. Waver: A model checking-based tool to verify web application design. *Electr. Notes Theor. Comput. Sci.*, 157(1):61–76, 2006.

[Con99] J. Conallen. Modeling web application architectures with UML. *Communications of the ACM*, 42(10):63–71, 1999.

[CZ04] Jessica Chen and Xiaoshan Zhao. Formal models for web navigations with session control and browser cache. In *ICFEM*, pages 46–60, 2004.

[dA01] Luca de Alfaro. Model checking the world wide web. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Proceedings of the Computer Aided Verification, 13th International Conference, CAV 2001, Paris, France, July 18-22*, volume 2102 of *Lecture Notes in Computer Science*, pages 337–349. Springer, 2001.

[dAHM01] Luca de Alfaro, Thomas A. Henzinger, and Freddy Y. C. Mang. MCWEB: A model-checking tool for web site debugging. In *Proceedings of the WWW Posters*, pages 86–87, 2001.

[DN06] Joumana Dargham and Sukaina Al Nasrawi. FSM behavioral modeling approach for hypermedia web applications: FBM-HWA approach. In *Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services*, pages 199–204. IEEE Computer Society, 2006.

[dOFT+01] de Oliveira, Maria Cristina Ferreira, Turine, Marcelo Augusto Santos, Masiero, and Paulo Cesar. A statechart-based model for hypermedia applications. *ACM Transactions on Information Systems*, 19(1):28–52, 2001.

[FHvLP00] Joachim Fischer, Eckhardt Holz, Martin von Löwis, and Andreas Prinz. Sdl-2000: A language with a formal semantics. In *Rigorous Object-Oriented Methods*, 2000.

[FLMM04] P. Fraternali, P. L. Lanzi, M. Matera, and A. Maurino. Exploiting conceptual modeling for web application quality evaluation. In *Proceedings of the 13th international conference on World Wide Web*, pages 342 – 343, 2004.

[Fra99] Piero Fraternali. Tools and approaches for developing data-intensive web applications: a survey. *ACM Computing Surveys*, 3:227–263, 1999.

[GFKF03] Paul T. Graunke, Robert Bruce Findler, Shriram Krishnamurthi, and Matthias Felleisen. Modeling web interactions. In Pierpaolo Degano, editor, *Proceedings of the Programming Languages and Systems, 12th European Symposium on Programming, ESOP 2003, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2003, Warsaw, Poland, April 7-11, 2003*, volume 2618 of *Lecture Notes in Computer Science*, pages 238–252. Springer, 2003.

[Gin02] Athula Ginige. Web engineering: managing the complexity of web systems development. In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, pages 721–729, 2002.

[GN04] Ekaterina Gorshkova and Boris Novikov. Use of statechart diagrams for modeling of hypertext. *Programming and Computer Software*, 30(1):47–51, 2004.

[Har87] D. Harel. Statecharts: A visual formalism for complex system. *Science of Computer Programming*, 8(3):231–274, 1987.

[HH06] Minmin Han and Christine Hofmeister. Modeling and verification of adaptive navigation in web applications. In *ICWE*, pages 329–336, 2006.

[HPS04] May Haydar, Alexandre Petrenko, and Houari A. Sahraoui. Formal verification of web applications modeled by communicating automata. In David de Frutos-Escrig and Manuel Núñez, editors, *Proceedings of the Formal Techniques for Networked and Distributed Systems - FORTE 2004, 24th IFIP WG 6.1 International Conference, Madrid Spain, September 27-30, 2004*, volume 3235 of *Lecture Notes in Computer Science*, pages 115–132. Springer, 2004.

[Jac02] Daniel Jackson. Alloy: A new technology for software modelling. In *Tools and Algorithms for the Construction and Analysis of Systems, 8th International Conference, TACAS 2002 April 8-12, 2002, Proceedings*, page 20, 2002.

[KBP00] M. Kitajima, M. H. Blackmon, and P. G. Polson. A comprehension-based model of web navigation and its application to web usability analysis. In *Proceedings of the HCI'00 Conference on People and Computers XIV*, Usability and System Evaluation, pages 357–374, 2000.

[KK03] Nora Koch and Andreas Kraus. Towards a common metamodel for the development of web appliactions. In *ICWE*, pages 497–506, 2003.

[KLH00] David Chenho Kung, Chien-Hung Liu, and Pei Hsia. An object-oriented web test model for testing web applications. In *COMPSAC*, pages 537–542, 2000.

[KR02] Chanwit Kaewkasi and Wanchai Rivepiboon. WWM: A practical methodology for web application modeling. In *Proceedings of the International Computer Software and Applications Conference*, pages 603–608. IEEE Computer Society, 2002.

[Kri05] Shriram Krishnamurthi. Web verification: Perspective and challenges. In María Alpuente, Santiago Escobar, and Moreno Falaschi, editors, *Proceedings of the First International Workshop on Automated Specification and Verification of Web Sites (WWV 2005), March 14-15, 2005 Valencia, Spain*, volume DSIC-II/03/05, pages 3–8. Departamento de Sistemas Informaticos y Computacion, Universidad Politecnica de Valencia, 2005.

[KZ06] Alexander Knapp and Gefei Zhang. Model transformations for integrating and validating web application models. In *Modellierung*, pages 115–128, 2006.

[LFP+02] Giuseppe A. Di Lucca, Anna Rita Fasolino, F. Pace, Porfirio Tramontana, and Ugo de Carlini. Comprehending web applications by a clustering based approach. In *Proceedings of the International Workshop on Program Comprehension*, pages 261–270. IEEE Computer Society, 2002.

[LHYT00] Karl R. P. H. Leung, Lucas Chi Kwong Hui, Siu-Ming Yiu, and Ricky W. M. Tang. Modeling web navigation by statechart. In *Proceedings of the International Computer Software and Applications Conference*, pages 41–47. IEEE Computer Society, 2000.

[LK04] Daniel R. Licata and Shriram Krishnamurthi. Verifying interactive web programs. In *Proceedings of the IEEE International Conference on Automated Software Engineering*, pages 164–173. IEEE Computer Society, 2004.

[LKHH00] Chien-Hung Liu, David Chenho Kung, Pei Hsia, and Chih-Tung Hsu. Structural testing of web applications. In *Proceedings of the International Symposium on Software Reliability Engineering*, pages 84–96. IEEE Computer Society, 2000.

[LKHH01] Chien-Hung Liu, David Chenho Kung, Pei Hsia, and Chih-Tung Hsu. An object-based data flow testing approach for web applications. *International*

*Journal of Software Engineering and Knowledge Engineering*, 11(2):157–179, 2001.

[Mil03] Craig S. Miller. Modeling web navigation: Methods and challenges. In Bamshad Mobasher and Sarabjot S. Anand, editors, *Proceedings of the Workshop on Intelligent Techniques for Web Personalization*, volume 3169 of *Lecture Notes in Computer Science*, pages 37–52. Springer, 2003.

[PMdO99] Fabiano Borges Paulo, Paulo Cesar Masiero, and Maria Cristina Fierreira de Oliveira. Hypercharts: Extended statecharts to support hypermedia specification. *IEEE Transactions on Software Engineering*, 25(1):33–49, 1999.

[PTdOM98] Fabiano B. Paulo, Marcelo Augusto S. Turine, Maria Cristina F. de Oliveira, and Paulo C. Masiero. XHMBS: A formal model to support hypermedia specification. In Kaj Grønbæk, Elli Mylonas, and III Frank M. Shipman, editors, *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia (HYPER-98)*, pages 161–170, New York, 1998. ACM Press.

[RC04] Peter Rob and Carlos Coronel. *Database Systems: Design Implementation And Management*. Course Technology, fifth edition edition, January 2004.

[Ric04] Filippo Ricca. Analysis, testing and re-structuring of web applications. In *Proceedings of the International Conference on Software Maintenance*, pages 474–478. IEEE Computer Society, 2004.

[Ric05] Filippo Ricca. Hyperlinks analysis in multilingual web applications. In *Proceedings of the International Workshop on Web Site Evolution*, pages 57–62. IEEE Computer Society, 2005.

[RT00] Filippo Ricca and Paolo Tonella. Web site analysis: Structure and evolution. In *Proceedings of the International Conference on Software Maintenance*, pages 76–86, 2000.

[RT01a] F. Ricca and P. Tonella. Analysis and testing of web applications. In *Proceedings of the 23rd International Conference on Software Engineering*, pages 25–36. IEEE Computer Society Press, 2001.

[RT01b] Filippo Ricca and Paolo Tonella. Building a tool for the analysis and testing of Web applications: Problems and solutions. In *Proceedings of the Tools and Algorithms for the Construction and Analysis of Systems Genova,Italy*, volume 2031, pages 373–388, 2 - 6 April 2001.

[SDM+05] Eugenio Di Sciascio, Francesco M. Donini, Marina Mongiello, Rodolfo Totaro, and Daniela Castelluccia. Design verification of web applications using symbolic model checking. In David Lowe and Martin Gaedke, editors, *Proceedings of the Web Engineering, 5th International Conference, ICWE 2005, Sydney,*

*Australia, July 27-29, 2005*, volume 3579 of *Lecture Notes in Computer Science*, pages 69–74. Springer, 2005.

[SDMP02] Eugenio Di Sciascio, Francesco M. Donini, Marina Mongiello, and Giacomo Piscitelli. Anweb: a sytem for automatic support to web application verification. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering , Ischia,Italy*, pages 609–616, July 14-19 2002.

[SDMP03] Eugenio Di Sciascio, Francesco M. Donini, Marina Mongiello, and Giacomo Piscitelli. Web applications design and maintenance using symbolic model checking. In *Proceedings of the European Conference on Software Maintenance and Reengineering*, pages 63–72. IEEE Computer Society, 2003.

[SFC98] P. David Stotts, Richard Furuta, and Cyrano Ruiz Cabarrus. Hyperdocuments as automata: Verification of trace-based browsing properties by model checking. *ACM Trans. Inf. Syst*, 16(1):1–30, 1998.

[SM03] Joe Abboud Syriani and Nashat Mansour. Modeling web systems using SDL. In Adnan Yazici and Cevat Sener, editors, *Proceedings of the Computer and Information Sciences - ISCIS 2003, 18th International Symposium, Antalya, Turkey, November 3-5, 2003*, volume 2869 of *Lecture Notes in Computer Science*, pages 1019–1026. Springer, 2003.

[SN02] P. David Stotts and Jaime Navon. Model checking cobweb protocols for verification of HTML frames behavior. In *Proceedings of the International World Wide Web Conference*, pages 182–190, 2002.

[SRBJ96] Daniel Schwabe, Gustavo Rossi, Barbosa, and Simone D. J. Systematic hypermedia application design with OOHDM. In *Proceedings of the Seventh ACM Conference on Hypertext*, Models of Hypermedia Design and Evaluation, pages 116–128, 1996.

[TR02] Paolo Tonella and Filippo Ricca. Dynamic model extraction and statistical analysis of web applications. In *Proceedings of the International Workshop on Web Site Evolution*, pages 43–52. IEEE Computer Society, 2002.

[TR04] Paolo Tonella and Filippo Ricca. A 2-layer model for the white-box testing of web applications. In *Proceedings of the International Workshop on Web Site Evolution*, pages 11–19. IEEE Computer Society, 2004.

[WBFP05] Marco Winckler, Eric Barboni, Christelle Farenc, and Philippe Palanque. What kind of verification of formal navigation modelling for reliable and usable web applications? In *Proceedings of the First International Workshop on Automated Specification and Verification of Web Sites (WWV'2005) , Valencia, Spain*, pages 33–36. LNCS, 14-15 mars 2005.

[Wik] The free encyclopedia Wikipedia. http://en.wikipedia.org/wiki/, last access november 2006.

[WO02] Ye Wu and Je Outt. Modeling and testing web-based applications. Technical report, George Mason University, 2002.

[WP03] Marco Winckler and Philippe A. Palanque. Statewebcharts: A formal description technique dedicated to navigation modelling of web applications. In *DSV-IS*, pages 61–76, 2003.

[ZBKK05] Gefei Zhang, Hubert Baumeister, Nora Koch, and Alexander Knapp. Aspect-oriented modeling of access control in web applications. In Mohamed Kandé, Dominik Stein, Omar Aldawud, Tzilla Elrad, Jeff Gray, and Jörg Kienzle, editors, *6th International Workshop on Aspect-Oriented Modeling*, March 2005.