Specification Patterns for Formal Web Verification

May Haydar [1], Houari Sahraoui [1], and Alexandre Petrenko [2]
[1] *Département d'informatique et de recherche opérationnelle, Université de Montréal*
CP 6128 succ. Centre-Ville, Montreal, Quebec, H3C 3J7, Canada
{Haidarma,Sahraouh}@iro.umontreal.ca
[2] *CRIM, Centre de recherche informatique de Montréal*
550 Sherbrooke West, Suite 100, Montreal, Quebec, H3A 1B9, Canada
Alexandre.Petrenko@crim.ca

Abstract

Quality assurance of Web applications is usually an informal process. Meanwhile, formal methods have been proven to be reliable means for the specification, verification, and testing of systems. However, the use of these methods requires learning their mathematical foundations, including temporal logics. Specifying properties using temporal logic is often complicated even to experts, while it is a daunting and error prone task for non-expert users. To assist web developers and testers in formally specifying web related properties, we elaborate a library of web specification patterns. The current version of the library of 119 functional and non-functional patterns is a result of scrutinizing various resources in the field of quality assurance of Web Applications, which characterize successful web application using a set of standardized attributes.

1. Introduction

Web development has become one of the largest and most important parts of the software industry, and yet little is known about how to fully ensure the quality of the developed applications [24]. In practice, currently web quality assurance is conducted informally. Meanwhile, during the last few years, formal approaches to verification and testing of software applications have been proven to be reliable means to insure their quality especially the behavioral, i.e., functional correctness. Likewise, several attempts have been proposed for formal verification and testing of WAs [2,4,5,8,25,27]. However, these approaches are often inapplicable by the web community because they do not exhaustively address the attributes and norms set by quality assurance specialists, and do not facilitate for web developers and testers the understanding of the mathematical foundation underlying the formal methods. In this paper, we propose a practical solution to the above stated problem which facilitates the acceptance of formal verification technology by the web community. We study the attributes that designate the good quality of such applications and propose a library of web temporal properties. The formalized requirements, i.e. specifications, relate mainly to the behavioral aspects of WAs based on user interactions and the way web resources are rendered. In this paper, we discuss the main attributes that determine the quality of WAs. We also present a non-exhaustive inventory of web quality requirements that can be formulated as formal properties of WAs. After surveying and analyzing several resources related to web quality assurance and usability, we were able to collect 119 quality requirements. In the context of a formal framework for run-time verification of WAs, see, e.g., [13,14], the web requirements once formalized can be verified against WA models using model checkers. However, to reduce the hurdle of learning the mathematical foundations underlying the temporal logic and model checking theory for web developers and testers, we map the collected quality requirements into Linear Temporal Logic (LTL). The formalized properties could serve as library of web specification patterns, which can be verified in a given WA.

The rest of the paper is organized as follows. Section 2 includes a discussion about the major web quality attributes. In Section 3, we present an overview of LTL. In Section 4, we discuss the pattern based approach to specify web properties, the classification we have used to categorize the web patterns, and the template used to represent them. In Section 5, we present the properties listed according to their categories and sub-categories, as well as examples of patterns for each sub-category. The full pattern system can be found in [9]. In Section 6, we present some results on the use of the web property patterns in

978-0-7695-3261-5/08 $25.00 © 2008 IEEE
DOI 10.1109/ICWE.2008.38

formal web verification. Section 7 includes the related work, and we conclude in Section 8.

2. Quality Assurance of Web Applications

Kan [18] and Offut [24] discuss the main quality attributes of WAs, which are explicitly dubbed as success criteria [18] for WAs. The most important quality attributes are found to be the frequently called reliability/functionality, usability, and security/privacy. These attributes embody the satisfaction and appeal of users, who if dissatisfied simply move away to another better quality WA. Other necessary attributes to high quality WAs are availability, scalability, maintainability, and performance. In this paper, we study the first three attributes, namely reliability/functionality, usability, and security/privacy. The main reason is that we concentrate on temporal web specifications related to the behavior (execution) of the WA itself rather than the specifications that relate to web servers and access load. This is due to the inaccessibility of web servers and their code in many cases, and most of the time, these attributes are evaluated mainly by analyzing server's logs, and measuring response time of the server. Thus, it is not possible to produce a formal model based on server's code or logs, and which can be used in model checking techniques. Therefore, the four latter attributes fall outside the scope of this paper. Additionally to the quality attribute dimension, we identify another dimension [20] which we believe is indispensable to the success of WAs, which we call the *stakeholder*. The stakeholder dimension introduces the idea that some requirements are specific to various stakeholders who have different interests in a given WA. While the above stated attributes are generic to WAs and are related to the satisfaction and appeal of the user, the stakeholder dimension is related to the satisfaction of the various stakeholders that are involved and have interest in the WAs. For example, while business owners are mostly interested in features which could result in directly achieving financial gains from the WA, advertisers are interested more in ensuring that their commercial advertisements properly appear in key web pages. Also, some WAs require particular specifications that are not covered by the existing quality requirements related to the above stated attributes. For instance, in WAs related to governments or political parties, the concern might be in carefully using some statements, phrases or words, which have to appear a certain number of times, in key web pages. Also, in online banking, where security is the most important feature among others, clients are given a limited number of unsuccessful attempts to login to their account.

The specifications to satisfy a particular need are pretty much subjective to the stakeholders themselves. For this reason, it is not possible to set a fix number of specific requirements related to this dimension. We collected a number of these requirements from the previous research work on formal verification of WAs [4,5,25,27], while we were able to infer some of them from browsing through particular WAs.

3. Linear temporal logic

Before we proceed to our approach, we give an overview of LTL. LTL (sometimes called PTL or PLTL) extends traditional propositional logic with temporal operators. LTL is used for property specification by major formal verification tools, for instance model checkers Spin and NuSMV. Also, LTL allows assertions about the temporal behavior of a system. An LTL formula φ has the following syntax:

$\varphi ::= p \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \,\mathtt{U}\, \varphi) \mid (\mathtt{G}\ \varphi) \mid (\mathtt{F}\ \varphi) \mid (\mathtt{X}\ \varphi)$

where p is an atomic proposition, `U` is the *until* operator, `G` (or □) is the *always* operator, `F` (or ◊) is the *eventually* operator, and `X` (or o) is the *next* operator. Informally, $\varphi\ \mathtt{U}\ \psi$ means that φ remains true until ψ becomes true. `G` φ means that φ is always true. `F` φ means that φ becomes true in a certain state. `X` φ means that φ is true in the next state.

LTL semantics is defined by Pnueli [26] over infinite sequences of states that correspond to infinite or non-terminating sequences of computations. LTL deals only with infinite behavior.

Let $M = (S, T, S_0, P, \mathcal{L})$ be a Kripke structure, where S is a set of states, $T \subseteq S \times S$ is a transition relation, $S_0 \subseteq S$ is a set of initial states, P is a set of atomic propositions, and $\mathcal{L}$ is a labeling function from S to the power set of P. An infinite state sequence $\pi = \langle s_0, s_1, \ldots\rangle$ is called a *path* (execution) of M if $s_0 \in S_0$, $(s_i, s_{i+1}) \in T$ for all i, $i \geq 0$. $\pi^i = \langle s_i, s_{i+1}, \ldots\rangle$ denotes the suffix of a sequence $\pi = \langle s_0, s_1, \ldots\rangle$ starting at s_i. Also, note that $\pi^0 = \pi$.

The semantics of LTL formulae on a sequence of states is defined as follows:

1. $\pi \vDash p \Leftrightarrow p \in \mathcal{L}(s_0)$,
2. $\pi \vDash \neg\varphi \Leftrightarrow \pi \nvDash \varphi$,
3. $\pi \vDash \varphi \wedge \psi \Leftrightarrow \pi \vDash \varphi$ and $\pi \vDash \psi$,
4. $\pi \vDash \mathtt{X}\ \varphi \Leftrightarrow \pi^1 \vDash \varphi$,
5. $\pi \vDash \mathtt{G}\ \varphi \Leftrightarrow$ for all i, $i \geq 0$, $\pi^i \vDash \varphi$,
6. $\pi \vDash \mathtt{F}\ \varphi \Leftrightarrow$ for some i, $i \geq 0$, $\pi^i \vDash \varphi$,

7. $\pi \models \varphi \, \mathrm{U} \, \psi \Leftrightarrow$ there exists an i, $i \geq 0$, such that $\pi^i \models \psi$ and for all j, $0 \leq j < i$, $\pi^j \models \varphi$.

The operator W is the *weak until* operator, such that $\varphi \, \mathrm{W} \, \psi = \varphi \, \mathrm{U} \, (\psi \vee \mathrm{G} \, \varphi)$.

Note that in the context of verification of WAs, a given WA is typically modeled by a Kripke structure, such that states represent web pages, transitions designate the links between the pages, and atomic propositions of each state are the attribute valuations in the corresponding web page. The page attributes are usually identified by the web user/tester.

While LTL is one of most popular formal notations for properties or requirements, it has certain limitations. For instance some properties, such as reachability of certain page cannot be described. However, reachability of page could be described in the negative way, that is as unreachability (G ¬p). Clearly, if page is not unreachable, it is reachable.

4. Pattern based Approach

Despite the automation of verification techniques, users of model checkers still must be able to specify system requirements in the specification language of the model checker, in particular, mastering the temporal logic theory. As an example, we present the following requirement for some types of WAs:
Incorrect login info is allowed for a limited number of times, and then login is forbidden.

To verify this requirement using LTL model checker, the web developer or tester has to translate it into the following LTL formula:
F *login* → F (*login* ∧ (X (¬ (*blocked*) ∨ (*relogin* ∧ X (¬ (*blocked*) ∨ (*relogin* ∧ X (¬ (*blocked*) ∨ X (*blocked* ∧ G (¬ (*login* ∨ *relogin*)))))))))))

such that *login* and *relogin* designate the login and re-login pages, and *blocked* designates the page indicating that the user is not allowed to login anymore after three trials, and that for instance he has to contact the company. Clearly, writing such formula is a daunting task for web developers and testers. Not only the formula is difficult to read and understand, but it also difficult to correctly formulate without having the expertise in LTL. To solve the above stated problem, we propose a pattern based approach to present the web specifications in an intuitive and easy to use manner. Dwyer et al. have developed the Specification Pattern System (SPS) [30,6,7], where a property specification pattern describes the essential structure of some aspect of a system's behavior and provides expressions of this behavior in a range of common formalisms [30]. However, we are not aware of any attempt to build a library of formal specification patterns that catalogue the requirements necessary to ensure high quality WAs. Also, although we employ patterns from the SPS, many of the web related formulae fall outside of the range of the SPS's patterns. In other cases, we should use several SPS patterns to represent a single web specification.

In the following, we present a library of web property specifications; we call it the Web Specification Pattern System (WeSPaS). This library is the result of surveying several resources related to web quality assurance [24,19,30,28], and web usability namely, IBM usability group [30], and various research work in the area of analysis and verification of WAs, including a Ph.D. research work [19] that has studied quality assurance of WAs. We have also deduced some requirements by noticing particularities related to some types of applications. In the analyzed resources, various requirements are developed for quality assurance of WAs. Among them, we identified 119 common requirements that can be formally specified and used in verification of WAs.

4.1. Categorization

To classify the web specifications, we do not keep the classical categorization of quality attributes. The reason is that some specifications identified as stakeholders specific overlap with the other quality attributes, while others do not fit in any of them. On the other hand, when analyzing the various quality requirements, it is possible to distinguish between requirements related to the ergonomics and design of web pages, and requirements related to the functionality of WAs which could cover a range of web pages of interest. For this reason, we propose a classification of the described requirements using two main categories:

Non-Functional. It groups the requirements that apply mainly to the design and ergonomics of web pages, which deal with standards related to links in pages, content management of pages, and navigation between pages. Therefore, we identify sub-categories:

1 Navigation and Links
2 Presentation and Content

Functional. It groups the requirements that relate to the functionality of WAs. We notice that many of them concern a wide range of WAs. On the other hand, we realize that e-commerce applications have specific quality requirements when it comes to the functionality, which do not necessarily apply to other types of WAs. Therefore, we identify three sub-categories: Reachability, and Security/Authentication and Trust, which concern WAs in general including e-commerce applications, and E-commerce which

concerns exclusively e-commerce applications. The groups are as follows:

1 Reachability
2 Security/Authentication and Trust
3 E-commerce
  3.1 Customer Service
  3.2 Product Info and Navigation
  3.3 Purchase Transaction

4.2. Template

In order to archive the web property patterns in a library, we need a template to characterize them in a systematic way. We propose the template which consists of:

- *ID*: a unique identifier for each pattern.
- *Pattern description*: an English description of the quality requirement.
- *Category*: the category and subcategory to which the pattern belongs.
- *Page Attributes*: the involved page attributes in the LTL formulation.
- *LTL Mapping*: the mapping of the quality requirement into LTL formula.
- *Comments*: additional information concerning the pattern and its formulation.
- *Source*: the source where the quality requirement of the pattern has been found.

The listed fields help the web developer and tester choose the properties of interest and provide the LTL formulation of the properties to verify. Also, the template's fields help in extending the library of patterns with new patterns. Note that each pattern is assigned a unique ID which encodes the first letters of the main category and subcategories followed by the number of the requirements.

5. Library of Web Property Patterns

In this section, we present the web property patterns constituting a library, categorized in their corresponding groups. Quality requirements are formalized as LTL properties that could be checked by a model checker given that the atomic propositions which constitute the formalized properties are attribute valuations existing in individual pages. We only present examples of full patterns for each category and sub-category. The complete library can be found in [9].

5.1. Non-Functional Patterns

In this section we present the non-functional patterns. They are classified into Navigation and Links, and Presentation and Content groups. They merely concern standard requirements for good design and ergonomics of WAs.

5.1.1. Navigation and Links. In this category, we have identified 36 pattern related to the navigational aspect of WAs as well as to the links present in web pages. Since many of the properties can be seen as both navigation and link property, we make them into one group. Below we present in Table 1 an example of Navigation and Links pattern number 17 whose description is: *there exist 1 or 2 links to Author or webmaster on every navigational path.*

Table 1. Navigation and Links pattern number 17.

ID	NFN17
Pattern description	There exist 1 or 2 links to Author or webmaster on every navigational path
Category	Non-functional – Navigation and Links
Page Attributes	*webmaster*: integer identification of webmaster page *webmaster _link*: Boolean indicating the presence for link to webmaster page
LTL Mapping	BoundedExistenceGlobally (*webmaster_link*)
Comments	
Source	Opquast

5.1.2. Presentation and Content. In this category we identify 25 patterns that relate merely to the design and ergonomics of each individual page, as well as to the content within the pages, such as counting certain objects, or insuring the presence/absence of certain objects or texts. Table 2 is an illustration of Presentation and Content pattern number 10.

Table 2. Presentation and Content pattern num. 10.

ID	NFC10
Pattern description	The number of a certain string/object should not appear more than a specific threshold
Category	Non-functional – Presentation and Content
Page Attributes	*num_string*: counts the number of a certain string or object
LTL Mapping	UniversalityGlobally (*num_string* <= *n*)
Comments	
Source	Opquast

5.2. Functional Patterns

Here we present the functional patterns, thus related to the functionality and expected behavior of WAs. They are classified into three main sub-categories: Reachability, Security/Authentication and Trust, and E-commerce.

5.2.1 Reachability. The patterns of this class concern WAs in general. These are 10 patterns related to the reachability of certain pages. As an illustration, we present in Table 3 the Reachability pattern number 2: *Home page is reachable from glossary without going through site map*.

Table 3. Reachability pattern number 2.

ID	FGR2
Pattern description	Home page is reachable from glossary without going through site map.
Category	Functional – Reachability
Page Attributes	*home*: integer identifying home page *site_map*: integer identifying site map *glossary*: integer identifying glossary page
LTL Mapping	Negation: ExistenceBetween (*glossary, home, site_map*)
Comments	To check this pattern, it is negated. The property formulated is "on all paths from glossary page to home page sitemap is present". If the result of verification gives a counter example, it means the model checker found at least a path from glossary page leading to home without going through sitemap. Then the original property is valid.
Source	Literature

5.2.2. Security/Authentication and Trust. We identified 12 patterns that are related to the security requirements of WAs. They include properties that ensure that secure pages are not accessed without proper authentication, or for specific types of applications, secure pages are accessed a certain number of times.

Table 4. Security pattern number 7.

ID	FGS7
Pattern description	Banking information is entered no more than once before submitting form
Category	Functional – Security/Authentication and Trust
Page Attributes	*Banking_info*: Boolean identifying the presence of fields for banking information *Submit*: identification of page where the submit button exists
LTL Mapping	F (*submit*) $\rightarrow$ ($\neg$ (*banking_info*) W (*banking_info* $\wedge$ X (G $\neg$ (*banking_info*) U *submit*)))
Comments	
Source	Newly introduced

To avoid intrusions or hackings, some properties ensure that the user is allowed to mistakenly login no more than a certain threshold. As an illustration, we present in Table 4 the Security pattern number 7: *Banking information is entered no more than once before submitting form.*

5.2.3. E-Commerce WAs. The class of E-commerce patterns comprises specification patterns that concern mainly e-commerce applications. We classify the patterns into the following sub-categories: Customer support, Product Info and Navigation, and Purchase Transaction. Note that this classification is manly adopted from the Usability Group at IBM [30].
For the Customer Support sub-category, we identify 13 patterns. For the Product Info and Navigation subcategory, 13 patterns, and for the Purchase Transaction sub-category, 10 patterns.

In Table 5, we present the pattern number 6 of the latter sub-category: *Credit card info is entered no more than once before submitting an order*.

Table 5. Purchase Transaction Pattern number 6.

ID	FEPT6
Pattern description	Credit card info is entered no more than once before submitting an order
Category	Functional – E-commerce – Purchase Transaction
Page Attributes	*Credit_card*: Boolean indicating the presence of fields requesting credit card info in pages *submit*: Boolean indicating the submit order page
LTL Mapping	F (*submit*) $\rightarrow$ ($\neg$ (*credit_card*) W (*credit_card* $\wedge$ X (G $\neg$ (*credit_card*) U *submit*)))
Comments	
Source	Newly introduced

6. Application

In [10,11,12,13] we have developed a formal framework for run time verification of WA based on Spin model checker [15]. A prototype tool is implemented to intercept requests and responses constituting execution traces [13,14] of a given WA and to infer an automata-based model translated to Promela, Spin's modeling language [15]. Through the graphical user interface of the tool, the user enters the page attributes of interest that need to be evaluated in the given WA.

We have conducted experiments on six WAs that are single window and multi display, where we verified properties on the inferred models. We verified properties from the WeSPaS which include reachability, security, navigation, and presentation properties. We list few of them:

1. Non-functional:
   - 1.1. Number of links in each display (single or multi) should not exceed a certain threshold (depends on size of application).
   - 1.2. Number of links in each display (single or multi) is balanced.
   - 1.3. Combinations of certain words/objects are absent
2. Functional:
   - 2.1. Secure pages are not reachable without authentication process.
   - 2.2. Promotions of certain products are only present either on the Home page or on Shopping pages.
   - 2.3. Secure pages are accessed exactly twice and each time with authentication.

The results of the verification of properties over the models of the WAs showed that many of the properties verified were violated. The medium-size applications are found to have the largest number of property violations, especially violations of reachability properties. On the other hand, multi display web applications especially with frames, irrespectively of their size or types of properties, were found to have a larger number of property violations than single window WAs, even with the simplest properties. Namely, most of the non-functional properties such as, *Combinations of certain words/objects are absent*, and *Number of links in each display (single or multi) is balanced*, though straightforward to check in single window WAs, were violated in most of the tested multi display WAs. This is due to the complex nature of those applications and the concurrent behavior that they exhibit.

7. Related Work

Web quality assurance has long been studied. Koyani [17] and Nielsen [22] suggest principles, guidelines, and recommendations to help developers in the design of web applications to ensure high quality WAs. Most of these proposals focus exclusively on WA usability aspects. In the measurement domain, Olsina [23] and Ivory [16] develop and classify a wide range of metrics. They also develop tools to automate, partially, the usability evaluation process [28]. Although the proposed principles, metrics, and tools improve the understanding of web quality and its evaluation, several considered quality requirements leave a room for subjective interpretation. In this paper, we studied the web quality assurance problem from a different perspective and offered an alternative means to web quality assurance using formal techniques.

Dwyer et al [6,7] present and analyze over 500 temporal properties, classified in the proposed specification pattern system [30]. The patterns constitute abstractions of specifications formulated for different formalisms in which such abstractions are not supported. The patterns are categorized into *Occurrence* and *Order*, and are defined on five scopes: *Globally*, *Before*, *After*, *Between*, and *AfterUntil*, which represent intervals/regions in which properties should be validated. The temporal properties surveyed in Dwyer's work refer to finite state verification of distributed systems, reactive systems, and timed systems, but not verification of WAs. While in our work, we not only analyze existing temporal web properties, but we study the essential requirements that satisfy quality attributes of WAs, which in turn we translate into temporal properties. Although we employ many of Dwyer's patterns at the specification level, we introduce a template to describe the web patterns in templates with a terminology used by the web community. We also utilize several patterns of the SPS to compose a single web pattern. Moreover, our WeSPaS include specifications using scopes of arbitrary subsets of states [10,11], which is not possible with SPS alone.

The only work we are aware of that uses patterns in Web related properties is that of Pereira et al [25] which specifies six design patterns for e-commerce applications classified as *construction* and *verification*. Those patterns are specific to the items to be sold and the assurance of correctness of their related transactions. The introduced patterns are specific only e-commerce applications and do not address other types of WAs. Also, unlike our work, they do not provide a library of specifications that helps in correctly choosing and instantiating a certain pattern for a particular e-commerce specification.

In conclusion, we believe that our pattern based approach to build a library of web formal specifications is novel and offers a practical solution to the complex problem of temporal logic formulation of web properties. To our knowledge, there have been no previous attempts to provide such a solution.

8. Conclusion and Future Work

In this paper, we developed a Web Specification Pattern System to reduce the hurdle for web users and testers in formally specifying web properties. We analyzed a range of resources in industry and academia referring to the quality attributes. We collected 119 requirements and built the WeSPaS [9], which is a repository of web patterns, categorized into Functional and Non-functional patterns. In the future, a tool can be developed and integrated, for instance, with our web analysis prototype tool [13] to allow the user to easily browse through the patterns, choose a particular pattern, and use its LTL formula for model checking. This opens the door to another direction of making the WeSPaS a public repository where other researchers and experts can add their contributions.

Acknowledgment. We would like to thank Sergiy Boroday for his valuable comments.

References

1. Boroday S, Petrenko A, Sing J, Hallal H. Dynamic Analysis of Java Applications for Multi Threaded Anti patterns. *3rd* International Workshops on Dynamics Analysis, *St-Louis*, MI, USA, May 17th, 2005.

2. Benedikt M, Freire J, Godefroid P. VeriWeb: Automatically Testing Dynamic Web Sites. *11th International World Wide Web Conference*, Hawai, U.S.A.
3. Clarke EM, Grumberg O, Peled DA. *Model Checking*. MIT Press, 2000.
4. De Alfaro L. Model Checking the World Wide Web. *13th International Conference on Computer Aided Verification*, Paris, France, July 2001.
5. Di Sciascio E, Donini FM, Mongiello M, Totaro R, Castelluccia D. Design Verification of Web Applications Using Symbolic Model Checking. *5th Int. Conference on Web Engineering*, LNCS 3579, Sydney, Australia, 2005.
6. Dwyer M, Avrunin GS, Corbett JC. Patterns in Property Specifications for Finite-state Verification. *21st Int. Conference on Software Engineering*, May, 1999.
7. Dwyer M, Avrunin GS, Corbett JC. Property Specification Patterns for Finite-state Verification. *2nd Workshop on Formal Methods in Software Practice*, March, 1998.
8. Dwyer MB and Clarke LA. Data flow analysis for verifying properties of concurrent programs. *2nd ACM SIGSOFT symposium on Foundations of software engineering*: 62-75, 1994.
9. Haydar M, Sahraoui H. WeSPaS: *A Specification Pattern System for Web Verification*. Technical Report [CRIM 07/10-17], Centre de Recherche Informatique de Montreal, October 2007.
10. Haydar M, Boroday S, Petrenko A, Sahraoui H. *Adding Propositional Scopes to Linear Temporal Logic*. Technical Report [CRIM 05/05-06], Centre de Recherche Informatique de Montreal, May 2005.
11. Haydar M, Boroday S, Petrenko P, Sahraoui H. Propositional Scopes in Linear Temporal Logic. *5th Int. Conference on Novelles Technologies de la Repartition*, Gatineau, Canada, August 30-September 1, 2005.
12. Haydar, M, Boroday S, Petrenko A, and Sahraoui H. Properties and Scopes in Web Model Checking. *20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005)*. Long Beach, California, USA, November 7-11, 2005.
13. Haydar M, Petrenko A, Sahraoui H. Formal Verification of Web Applications Modeled by Communicating Automata. *24th IFIP WG 6.1 IFIP Int. Conference on Formal Techniques for Networked and Distributed Systems*, LNCS, Spain, September 2004, 3235:115-132.
14. Haydar M. Formal Framework for Automated Analysis and Verification of Web-based Applications. *19th IEEE Int. Conference on Automated Software Engineering*, Linz, Austria, September 20-24, 2004
15. Holzmann GJ. *The Spin Model Checker, Primer and Reference Manual*. Addison-Wesley, 2003.
16. Ivory, M. *An Empirical Foundation for Automated Web Interface Evaluation*, Doctoral Thesis, 2001.
17. Koyani SJ, Bailey RW, Nall JR. *Research-Based Web Design & Usability Guidelines*. National Institutes of Health, 2003.
18. Kan S. *Metrics and Models in Software Quality Engineering*, 2nd ed. Pearson, 2002.
19. Malak G, Sahraoui H. Badri L, and Badri M. Modeling Web-Based Applications Quality: a Probabilistic Approach. *7th International Conference on Web Information Systems Engineering (WISE)*, 2006.
20. Malak G, Badri L, Badri M, Sahraoui H. Towards a Multidimensional Model for Web- Based Applications Quality Assessment. *5th International Conference on E-Commerce and Web Technologies*, Spain, LNCS Vol. 3182. Springer-Verlag, 316-327, 2004.
21. Millerand F, Martial O. *Guide pratique de conception et d'évaluation ergonomique de sites Web*. Montréal, Centre de recherche informatique de Montréal, 2001. [CRIM-01/08-21].
22. Nielsen, J. *Designing Web Usability : The Practice of Simplicity*. New Riders Publishing, 2000.
23. Olsina, L. Web-site Quality Evaluation Method : A Case Study on Museums. *Proceedings of ICSE 99 – 2nd Workshop on Software Engineering over the Internet*, 1998.
24. Offutt J. Quality attributes of Web software applications. IEEE Software, 19(2): 25-32, 2002.
25. Pereira A, Song M, Gorgulho G. *The Formal-CAFE methodology and model checking patterns in the specification of e-commerce systems*. Electronic Commerce Research, 6: 265-303, Springer Verlag, 2006.
26. Pnueli, A. The Temporal Logic of Programs. *18th IEEE Symposium on Foundations of Computer Science*, 1977, 46-57.
27. Stotts PD, Cabarrus CR. Hyperdocuments as Automata: Verification of Trace-Based Browsing Properties by Model Checking. *ACM Transactions on Information Systems*, January 1998, 16(1): 1-30.
28. Shubert P, Dettling W. *Extended Web Assessment Method (EWAM) - Evaluation of Electronic Commerce Applications from the Customer's Viewpoint*. 35th Hawaii International Conference on System Sciences, 2002.
29. Temesis - Qualité, conformité, et accessibilité des sites Internet. *Opquast: Quality Best Practices for On-line Services*. 2005, from http://en.opquast.com/.
30. *The Specification Patterns System*. 2002, from http://patterns.projects.cis.ksu.edu/.
31. IBM Usability. *Web Design Guidelines*. 2005, from http://www-306.ibm.com/ibm/easy/eou_ext.nsf/publish/611.