INFORMATICA, 2012, Vol. 23, No. 2, 191–201
© 2012 *Vilnius University*

Comparing Real and Intended System Usages: A Case for Web Portal

Jérémy BESSON, Audronė LUPEIKIENĖ, Viktor MEDVEDEV
Akademijos 4, LT-08663 Vilnius, Lithuania
e-mail: contact.jeremy.besson@gmail.com

Received: January 2011; accepted: October 2011

Abstract. Regarding the complexity of actual software systems, including web portals, it is becoming more and more difficult to develop software systems such that their real usage will satisfy their intended usage. To tackle this problem, we can compare the a priori assumptions about how the system should be used with the actual user behavior in order to decide how the system could be improved. For this aim, we propose to employ the same formalism to express the intended usage, the web portal model and the real usage extracted from system usage traces by data mining algorithms. Inspired from BioCham, we propose to use temporal logic and Kripke structure as such a common formalism.
Keywords: intended usage, real usage, web portal model, linear temporal logic, pattern mining.

1. Introduction

Regarding the complexity of actual software systems, including web portals, there is a huge gap between how developers want the software system to be used and how it is used in practice. The functionalities that a software should provide is usually well studied but there are only the building blocks of the whole system. Requirement specifications typically contain both functional requirements (what does a system) and non-functional requirements (characteristics that a software system must exhibit while performing its task). Both kinds of requirements need to be taken in account when designing a software system.

As a result, it is not uncommon that software systems suffer from quality problems that prevent or inhibit their use, e.g., difficulty to find the right information, poor navigation or inappropriate display of information. End-users have their own perspectives, goals, experience and skills, and it is very difficult, if not impossible, to fully guess the users' expectations at the moment that the software system is designed. However, frequently these assumptions hold only partly and are defined only partially. However, it is not easy to make these assumptions because of great variety of users. First, they have different preferences and features including background knowledge, reasoning styles, and experience. Second, their needs and goals are different. The web portal visitors seek to accomplish tasks (doers), to get information (viewers) or relaxation (readers) (Zeldman, 2001). Third, the type and amount of information the users are interested in strongly differ. Therefore, one must be capable to compare the a priori assumptions with the actual user behavior in order to decide how the system should be improved. An a posteriori evaluation of the difference between the intended and the real usage of a software system is a critical issue for providing the user with the best software system use experience.

In this article, we focus on web portal. In general, a web portal is a web site which is an entry point to other websites. It provides users with a single point of access to information and services (e.g., e-mail and news). We consider the subclass of web portals and discuss only the problems related to the specification and design of corporative portals. A corporate portal, also known as an enterprise portal, enables the collecting, sharing, and dissemination of information throughout the intranet, extranet and Internet. It can be thought of as a framework for integrating people, information and processes across organizational boundaries. The corporate portal design process should start with determining and understanding users. All design decisions directly depend on the audience the portal intends to serve. However, assumptions about how the system should be used are usually not formulated in an explicit way.

To allow automatic reasoning and verification on the indented and real usage, one needs to represent the system in a formal way. A formal representation of the real system called "Model" can be used to reproduce the possible real usages of the system. From the real system, usage traces can be obtained. Intended usage must be defined formally as a sub-set of the possible usages. The question to be answered from these three elements, i.e., web portal model, real usage and intended usage, is: do the real usages satisfy the intended usages providing the web portal model? To tackle this problem, we consider as a promising approach to employ the same formalism to express the intended usage, the web portal model and the frequent real usage patterns extracted from the system usage traces by data mining algorithms. This allows to automate the verification whether the frequent real usage patterns satisfy the intended usage in the web portal model. Inspired from BioCham (Chabrier-Rivier *et al.*, 2004, 2005; Calzone *et al.*, 2005a, 2005b), we propose to use temporal logic and Kripke structure as such a common formalism. This article extends the work of Besson *et al.* (2010) providing a new insight of the proposed method as well as a new language to express the model and the intended usage.

The paper is organized as following. Section 2 presents BioCham that inspired our work. Then, in Section 3, we present a state-of-the-art on testing and verifying web portals using model checking as well as the linear Temporal Logic and Kripke structure formalisms that are employed along this article. Section 4 presents our method for modeling a web portal. The model to define intended usage is presented in Section 5. Section 6 presents how real usages are extracted. Afterwards, we describe how intended and real usage are compared (see Section 7). Finally, we briefly conclude the work in Section 8.

2. Getting Inspired of BioCham

Biocham (Biochemical Abstract Machine; Chabrier-Rivier *et al.*, 2004, 2005; Calzone *et al.*, 2005a, 2005b) is a formal environment for modeling biological networks. It provides (1) a rule-based language for modeling biochemical systems with patterns and constraints, (2) a simple simulator of the rule-based model, (3) a query language based on CTL (Computation Tree Logic) and (4) an interface to the NuSMV symbolic model checker for automatically evaluating CTL queries.

An important problem in biology is to model bio-molecular interaction maps, e.g., protein-protein and protein-DNA interaction networks. In the literature, information of two natures can be found about these maps: local interactions of the involved biological elements and global behaviour of the interaction map. Examples of local interactions are (1) if protein P1 and protein P2 are present then they can form a new complex named P1-P2 and (2) protein P1 can be degraded (disappear). They represent some kind of transitions between states that describe the presence or absence of the proteins (more precisely, the quantity of the proteins). Global behaviour describes the possible evolutions of the whole system. For example, some states are reachable while others not, some properties are invariant, e.g., the presence of the protein P1, or the presence of a cycle.

In this context, biologists seek to be able to express their knowledge in terms of both local interactions and global behaviour, and then to check if they are compatible, i.e., if the local interactions lead to the known global behaviour. If it is not the case, then some of the local interactions can be erroneous and/or some others can be missing. The goal here is to be able to iteratively remove/add some local interactions and thereby enrich the model, in order to converge toward a model that exhibits the needed global behaviour.

In Biocham, the local interactions are defined by the means of a Kripke structure (see Section 3) that models the presence or absence of the different biochemical compounds over time and are represented as rules. Rules in Biocham are asynchronous in the sense that only one reaction rule is applied at a time. That Kripke structure is a non-deterministic transition system where the temporal evolution of the system is modeled by the succession of the transition steps (local transitions or rules), and the different possible behaviors of the system are obtained by the non-deterministic choice of reactions. Biocham supports the use of CTL (Computation Tree Logic) as a query language for querying the temporal properties of models (see Section 3). The known global behaviours of the biochemical interaction maps are then expressed in the form of CTL formulas. The state-of-the-art symbolic model checker NuSMV is used to verify whether the CTL formulas are verified on the Kripke structure, i.e., if the provided local transitions lead to the expected behaviour. Figure 1 presents the BioCham tool.

This work strongly inspired our work, especially for its ability to combine known and expected knowledge, to encompass a system model, local interactions or actions and a global behavior, and finally to consider time as the main dimension.

3. Testing and Verifying Web Portals Using Model Checking

Verification and testing of web portals has received significant attention in recent years (Alalfi *et al.*, 2007; Di Sciascio *et al.*, 2005; Han and Hofmeister, 2006). Web portals include web pages with different kinds of information such as texts, images, and forms. Web pages can be static or dynamic. Dynamic pages are the ones that change in an automatic way. Dynamic pages can change every time they are loaded and they can change

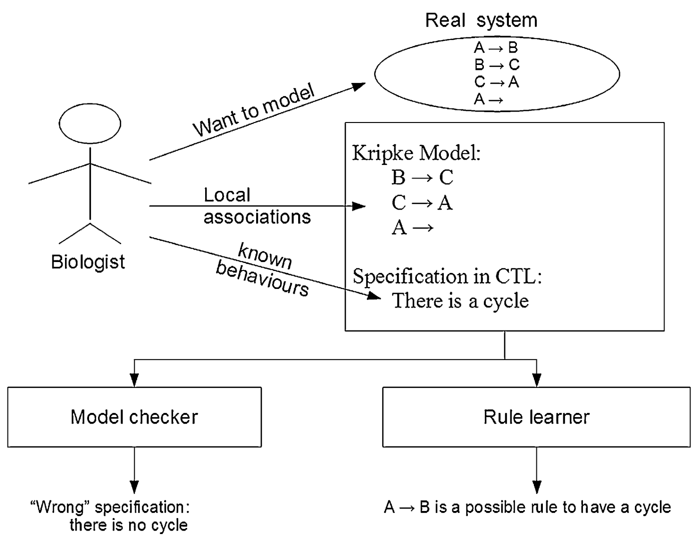

Fig. 1. Biochemical Abstract Machine – BIOCHAM.

their content based on what user does, i.e., clicking on some text or image. Researchers are still trying to find effective ways to model and test web portals. In Alalfi *et al.* (2007), methods are described and proposed to gather and process different properties related to the structure of web portals, navigation, behavior and content. Web applications development involves a number of new languages, programming and technologies models, that are used to implement different applications having high quality requirements. Web applications are sophisticated interactive programs with components that are integrated in novel ways. Modeling, analyzing and testing these applications present a number of new challenges to researchers.

Model checking (Clarke *et al.*, 1999) is a formal verification technique. It has been shown to be especially useful for verifying properties, and identifying bugs related to process schedules. Model checking is a formal technique for automatically verifying correctness properties of finite-state systems. Model checking for web portals starts with a model described by the user, and discovers whether or not properties used by the user are valid.

Temporal logic is one of the symbolic system to perform model checking. Temporal logic extends propositional logic that describes the states, with operators for reasoning over time and non-determinism. Temporal logic is dedicated to express time constraints or properties of a system (Pnueli, 1992; Dixon *et al.*, 2007; Zhou, 1987; Rossi *et al.*, 2004; Emerson, 1996). In this article, we will consider LTL (Linear Temporal Logic) as a convenient formalism to express dynamic properties of our system, especially for intended usage and real usage.

Several temporal operators are introduced in LTL w.r.t. propositional logic: X ϕ means ϕ is true at the next transition, G ϕ means ϕ is always true, F ϕ means ϕ finally true, and

ϕ U ψ means ϕ is always true until ψ becomes true. A LTL formula is true if every path of the Kripke structure, starting from a starting node, satisfies the LTL formula.

LTL formulas can be can be verified on a Kripke Structure which can be represented as a finite-state automaton. Such an automaton is defined with a set of discrete variables (Boolean, enumeration, bounded integer $[a \ldots b]$). An instantiation of all the variables defines a state in the automaton. Transitions between the states (variable instantiations) are also defined. The semantics of the Kripke structure is given by the set of its paths, i.e., the set of infinite sequences of states of the Kripke structure from the start node.

Interestingly, with such a formalism (LTL and kripke structure), we can both define and reason on a web portal model defined as a Kripke structure, and on the intended usage and real usages defined as LTL formulas.

Several verification tools have been developed for system analysis based on different formal models with respect to the web portal specifications. Specifications are expressed in a logical formalism. In Di Sciascio *et al.* (2005), a mathematical model of a web application partitioning the usual Kripke structure into links, pages and actions has been proposed. Verification is performed using the Symbolic Model Verifier (SMV). In another approach (Stotts *et al.*, 1998), an automaton is presented to describe the structure of the links in a hypertext and a branching temporal logic HTL is proposed to describe the sequence of transitions between states in the automaton.

Haydar *et al.* (2004) present an approach to formally model web applications for the purpose of verification and validation using model checking. They use the dynamic approach by executing the application under test (navigation and form filling), and observing the external behavior of the application.

Kung *et al.* (2000) propose a model that extends traditional test models, such as data flow graph and finite state machines to web applications for capturing their test-related artifacts. Based on the test model proposed in the paper, test cases for validating web applications can be derived automatically. Finite-state machines were employed for testing web applications modeling subsystems of the web applications (Andrews *et al.*, 2005).

A model for the verification of properties of web portals is presented in Flores *et al.* (2008). In this work, a website is defined as a collection of web pages which are semantically connected in some way. Flores *et al.* (2008) present the logic that is used to specify properties of the websites, and illustrate the kinds of properties that can be specified and verified using model-checking. It is shown how to verify properties of a web application that are specified using Linear Temporal Logic (Manna and Pnueli, 1992), and how to apply model-checking in order to check the properties of the studied system in practice. The relation between the proposed model and Kripke structure is established, what makes possible to apply the model-checking technique to web applications. The most important issue in such relation is the definition of the labeling function of the Kripke structure.

Navigation on a web application can be defined as the possible sequences of web pages that a user visit during her/his session. Navigation models are useful for clarifying requirements and specifying users' behavior. Many web applications now incorporate adaptive navigation, where the next page also depends on user type and on which pages she/he has visited before. A formal approach that uses Statecharts is presented to formally model adaptive navigation (Han and Hofmeister, 2006). This method can help to understand how important properties of a navigation model are verified using existing model-checking tools.

4. Web Portal Model

We now propose a formal grammar to define a web portal model. Let us remind that a formal grammar G is a 4-tuple $G = (N, \Sigma, P, S)$ where N is a finite set of non-terminal symbols, Σ is a finite set of terminal symbols that is disjoint from N, P is a finite set of production rules and S is the starting point of the grammar. We define the grammar $G_{wpu} = (N, \Sigma, P, S)$ to model a web portal where:

- $N =$ { Start, States, State, VariableValue, Instantiation, Value, Transistions, Transition, StateName }
- $\Sigma =$ { NL, @, =, ⇒, ; , :, $0 \ldots 9$, a… Z, true, false } where NL denotes the new line
- $S =$ Start
- $P =$

  Start → States NL Transitions
  States → State NL State | State
  State → StateName : VariableValue
  VariableValue → Instantiation; VariableValue | Instantiation
  Instantiation → @[a–Z] + = Value
  Value → true | false | [0–9]$^+$ + | [a–Z]$^+$
  Transitions → Transition NL Transition | Transition
  Transition → StateName ⇒ StateName | StateName ⇒
  StateName : VariableValue
  StateName → S[1–9][0–9]*

A sentence of G_{wpu}:

- "S1: @document = P1; @Goal = G" means that S1 is a state such that the variable @document is equal to "P1" and the goal "G" is true. Thereby we state that the user has the goal "G" when she/he arrives to the page "P1".
- "S2: @document = P2; @category = C1" means that S2 is a state such that the variable @document is equal to "P2" and the current page's category is "C1".
- "S1 → S2: @userIslogged = true" means that the user can go from state S1 to state S2 if the variable @userIslogged is equal to true, i.e., if the user is logged in.

A web portal model defined with the grammar G_{wpu} can be automatically transformed into a Kripke structure.

5. Intended Usage

In addition to traditional LTL formulas on Kripke structure that defines the web portal model, we introduce variables in the LTL formulas in order to be able to express certain constraints, such as, e.g., "at least one goal is reached", without writing a complicated LTL formula as "F(@Goal1 = true ∧ F @Goal = false) ∨ F(@Goal2 = true ∧ F @Goal2 = false) ∨ F(@Goal3 = true ∧ F @Goal3 = false) ∨ . . . ". These variables that we add in the LTL constraint will be replaced by a set of discrete variables of the Kripke structure. Such variable replacement must be effectuated so that each LTL constraint variable is replaced by a different variable of the Kripke structure. We define a LTL formula with variables to be true if (1) the formula is true for all the possible replacements of the variables or (2) the formula is true for at least one of the possible replacements of the variables. To distinct these two types of formulas we add two operators: "Vall" and "Vone". We place these operators in front of the LTL formulas containing variables in order to state, respectively, that "all the instantiations must be true" (Vall) or "at least one instantiation must be true" (Vone). Consider, for example, a web portal model defined by a Kripke structure that contains three variables @Goal1, @Goal2 and @Goal3, and the intended usage expressed by the constraint "at least one goal is reached". This can we written by the following LTL formula with variables: "Vone {@Goal1, @Goal2, @Goal3} F (Goal = true ∧ F (Goal = true))". After the step of LTL formula's variables replacement with the Kripke structure variables, this formula is transformed into "F (@Goal1 = true ∧ F (@Goal1 = false)) ∨ F (@Goal2 = true ∧ F (@Goal2 = false)) ∨ F (@Goal3 = true ∧ F (@Goal3 = false))". Consider another example of the intended usage "user can only have one goal at a time", which can be written with the following LTL formula with variables "Vall {@Goal1, @Goal2, @Goal3} ! F(G1 = true ∧ G2 = true)", which, after the replacement step, is transformed into "! F(@Goal1 = true ∧ @Goal2 = true) ∧ ! F(@Goal2 = true ∧ @Goal3 = true) ∧ ! F(@Goal1 = true ∧ @Goal3 = true)". Every LTL formula's variable must be replaced by a different Kripke variable. In the previous example, G1 and G2 must refer to different Kripke variables and cannot be replaced with the same variable, i.e., ! F(@Goal1 = true ∧ @Goal1 = true) is an incorrect replacement.

We propose the grammar $G_{IU} = (N, \Sigma, P, S)$ to express intended usage:

- N = {Start, Formula, LTL, LTLOper, Expr, Value, Names}
- Σ = { =, ;, ⇒, ∧, ∨, !, {, }, @, "X", "F","G", "U", "V", "S","T" }
- S = Start
- P =

  Start → Formula; Start | Formula
  Formula → Vone {Names} LTLOper LTL | Vall { Names} LTLOper LTL | LTLOper LTL
  LTL → LTL ∧ LTL | LTL ∨ LTL | Expr ⇒ LTL | LTLOp LTL | Expr | Expr U Expr | Expr V Expr | Expr S Expr | Expr T Expr | ! LTL
  LTLOper → X | F | G | ! LTLOper
  Expr → @[a–Z] + = Value
  Value → true | false | [0–9] +| [a–Z] +
  Names → @[a–Z] +, Names | @[a–Z] +

Following are several examples of intended usage expressed with the grammar G_{IU}:

- Never go three times to the home page: ! F(@document = home ∧ X F(@document = home ∧ X F(@document = home)))

- Reach at least one goal of @Goal1, @Goal2 and @Goal3:
  Vone {@Goal1,@Goal2,@Goal3} F(G1 = true $\wedge$ F(G1 = false))
- User can have only one goal: Vall {@Goal1,@Goal2,@Goal3}
  ! F(G1 = true $\wedge$ G2 = true)

6. Extracting Frequent Usage Patterns

Frequent usage patterns can be extracted form event files (log files) using sequential pattern mining algorithms. Some frequent web portal usage patterns can be revealed by extracting sequential patterns as episode rules (Meger and Rigotti, 2004; Mannila *et al.*, 1997; Min and Kai, 2004), frequent sub-strings (DeRaedt *et al.*, 2002; Dan Lee and De Raedt, 2004; Weiner, 1973; Ukkonen, 1995) and frequent sub-sequences (Agrawal *et al.*, 1995; Ayres *et al.*, 2002). Interestingly, these frequent sequential patterns can be transformed into LTL formulas. For example, a frequent sub-string $[S_1, S_2, S_3, \ldots]$ which is, in our case, a string of successive clicks that appears in at least $x\%$ of the log sessions can be expressed in LTL with the following formula: F(@document $= S_1 \wedge$ X(@document $=$ $S_2 \wedge$ X(@document $= S_3 \wedge$ X($\ldots$)))). A frequent sub-sequence $[S_1, S_2, S_3, \ldots]$ which is a sequence of not necessary successive clicks that appears in at least $x\%$ of the log sessions and such that the consecutive clicks of the extracted sequence appear in the log session within a window of maximal size y, can be expressed with the following LTL formula: F(@document $= S_1 \wedge$ X F(@document $= S_2 \wedge$ X F(@document $= S_3 \wedge$ X F($\ldots$)))). An episode rule $[S1_1, \ldots, S1_n \Rightarrow S2_1, \ldots, S2_m]$ which is a couple of sequences (S1,S2) of clicks such that if the sequence S1 is present in a log session then S2 is also present afterward with a confidence of at least $x\%$ and it appears in at least $y\%$ of the log session, can be expressed with the following LTL formula: G(@document $=$ $S1_1 \wedge$ X(@document $= S1_2 \wedge \ldots \wedge$ @document $= S1_n \wedge$ X F(@document $= S2_1 \wedge$ X(@document $= S2_2 \wedge \ldots \wedge$ X(@document $= S2_m$])))) $\vee$! (@document $= S1_1 \wedge$ X(@document $= S1_2 \wedge \ldots \wedge$ @document $= S1_n$))).

7. Comparing Real and Intended Usage Providing the Web Portal Model

Until now, we have described (1) how to model a web portal using a Kripke structure, (2) how to specify intended usage in LTL and (3) how to extract frequent usage patterns from usage traces by the means of Data Mining algorithms. To verify that the extracted frequent usage patterns are coherent w.r.t. the intended usage, defined with the language G_{IU} providing a web portal model (G_{wpu}), we check if it exists a path in the Kripke structure (G_{wpu}) that satisfies all the intended usages (G_{IU}) and the LTL formulas obtained from the frequent usage patterns. Thereby, one can verify if the frequent usage patterns are coherent w.r.t. the intended usage. For doing that, we generate and check the LTL formula "! (C1 $\wedge$ C2 $\wedge \ldots \wedge$ Cn $\wedge$ Cp)" where C1, C2, $\ldots$, Cn are the LTL formulas specifying the intended usage, and Cp is the LTL formula obtained from a frequent usage pattern. The negation on the LTL formula comes from the fact that a LTL formula is true if it is true in all its initial states and for all the paths of the Kripke structure. But our aim is to check if it exists at least one path that satisfies the LTL formula. So, we use the idea that if a LTL formula $!L_1$ does not hold on all the computation paths then it exists a computation path such L_1 holds.

A frequent usage pattern is incoherent w.r.t. the intended usage, if no path can follow the frequent usage pattern while satisfying the intended usage. The frequent usage pattern is incoherent w.r.t. the intended usage, if the generated LTL formula is true. To check whether the generated LTL formula is true over the Kripke structure model of a web portal we use the NuSMV symbolic model checker.

8. Conclusions

We present a method to compare intended usage and real usage in web portal that use the same formalism (LTL and Kripke structure) to represent the web portal model, intended usage and real usage. Data Mining techniques are used to extract usage patterns from the log files.

Acknowledgments. This research was funded by a grant (No. TAP-43/2010) from the Research Council of Lithuania and by the French–Lithuanian bilateral program EGIDE PHC “Gilibert” under the grant 19976RM.

References

Agrawal, R., Srikant, R. (1995). Mining sequential patterns. In: *Proceedings of the Eleventh International Conference on Data Engineering (ICDE'95)*. IEEE Computer Society, pp. 3–14.

Alalfi, M.H., Cordy, J.R., Dean, T.R. (2007). A survey of analysis models and methods in website verification and testing. In: *Proceedings of the 7th International Conference on Web Engineering (ICWE)*, pp. 306–311.

Andrews, A.A., Offutt, J., Alexander, R.T. (2005). Testing web applications by modeling with FSMs. *Software and System Modeling*, 4(3), 326–345.

Ayres, J., Flannick, J., Gehrke, J., Yiu, T. (2002). Sequential pattern mining using a bitmap representation. In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, pp. 429–435.

Besson, J., Mitasiunaite, I., Lupeikiene, A., Boulicaut, J.-F. (2010). Comparing intended and real usage in web portal: temporal logic and data mining. In: *Business Information Systems, 13th International Conference (BIS 2010). Lecture Notes in Business Information Processing*, Vol. 47, pp. 83–93.

Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S. (2005a). A machine learning approach to biochemical reaction rules discovery. In: *Proceedings of Foundations of Systems Biology and Engineering (FOSBE'05)*, pp. 375–379.

Calzone, L., Chabrier-Rivier, N., Fages, F., Gentils, L., Soliman, S. (2005b). Machine learning bio-molecular interactions from temporal logic properties. In: *Proceedings of the Third International Conference on Computational Methods in Systems Biology (CMSB'05)*.

Chabrier-Rivier, N., Fages, F., Soliman, S. (2004). The biochemical abstract machine BIOCHAM. In: *Proceedings of the Second International Workshop on Computational Methods in Systems Biology (CMSB'04)*. Springer, Berlin, pp. 172–191.

Chabrier-Rivier, N., Fages, F., Soliman, S., Calzone, L. (2005). *Learning Transition Rules from Temporal Logic Properties*. Research report, INRIA, 5543.

Clarke, E., Grumberg, O., Peled, D. (1999). *Model Checking*. MIT Press, Cambridge.

Dan Lee, S., De Raedt, L. (2004). An efficient algorithm for mining string databases under constraints. In: *Proceedings of the Third international conference on Knowledge Discovery in Inductive Databases (KDID'04)*. Springer, Berlin, pp. 108–129.

De Raedt, L., Jaeger, M., Dan Lee, S., Mannila, H. (2002). A theory of inductive query answering. In: *Proceedings of the IEEE International Conference on Data Mining (ICDM'02)*, pp. 123–130.

Di Sciascio, E., Donini, F.M., Mongiello, M., Totaro, R., Castelluccia, D. (2005). Design verification of web applications using symbolic model checking. In: Lowe, D., Gaedke, M. (Eds.), *Proceedings of the Web Engineering, 5th International Conference (ICWE 2005). Lecture Notes in Computer Science*, Vol. 3579, pp. 69–74.

Dixon, C., Fisher, M., Konev, B., Lisitsa, A. (2007). *Efficient First-Order Temporal Logic for Infinite-State Systems*. Computing Research Repository, Cornell University.

Emerson, E.A. (1996). Automated temporal reasoning about reactive systems. *Logics for Concurrency: Structure Versus Automata*. Springer, Berlin. *Lecture Notes in Computer Science*, Vol. 1045, pp. 41–101.

Flores, S., Lucas, S., Villanueva, A. (2008). Formal verification of websites. *Journal Electronic Notes in Theoretical Computer Science (ENTCS)*, 200(3), 103–118.

Han, M., Hofmeister, C. (2006). Modeling and verification of adaptive navigation in web applications. In: *Proceedings of the 6th International Conference on Web Engineering (ICWE'06)*, pp. 329–336.

Haydar, M., Petrenko, A., Sahraoui, H.A. (2004). Formal verification of web applications modeled by communicating automata. In: *Proceedings of the Formal Techniques for Networked and Distributed Systems – FORTE 2004, 24th IFIP WG 6.1 International Conference*, Spain. *Lecture Notes in Computer Science*, Vol. 3235, pp. 115–132.

Kung, D.C., Liu, C.-H., Hsia, P. (2000). An object-oriented web test model for testing web applications. In: *Proceeimgs of the Twenty-Fourth Annual International Computer Software and Applications Conference (COMPSAC'2000)*, pp. 537–542.

Manna, Z., Pnueli, A. (1992). *The Temporal Logic of Reactive and Concurrent Systems – Specification*. Springer, New York.

Mannila, H., Toivonen, H., Verkamo, A.I. (1997). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1, 259–289.

Méger, N., Rigotti, C. (2004). Constraint-based mining of episode rules and optimal window sizes. In: *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04)*, pp. 313–324.

Min, Q., Kai, H. (2004). Frequent episode rules for internet anomaly detection. In: *Proceedings of the Network Computing and Applications, Third IEEE International Symposium (NCA'04)*. IEEE Computer Society, pp. 161–168.

Pnueli, A. (1992). System specification and refinement in temporal logic. In: *Proceedings of Foundations of Software Technology and Theoretical Computer Science*. Springer, Berlin, pp. 1–38.

Rossi, C., Enciso, M., Mora, A. (2004). A first order temporal logic for behavior representation. In: *Advances in Artificial Intelligence*, Springer, Berlin. *Lecture Notes in Computer Science*, Vol. 3315, pp. 408–418.

Stotts, P.D., Furuta, J.C., Ruiz, C. (1998). Hyperdocuments as automata: verification of trace-based browsing properties by model checking. *ACM Transactions on Information Systems*, 16(1), 1–30.

Ukkonen, E. (1995). On-line construction of Suffix Trees. *Algorithmica*, 14(3), 249–260.

Weiner, P. (1973). Linear pattern matching algorithm. In: *Proceedings of the 14th IEEE Symposium Switching and Automata Theory*, pp. 1–11.

Zeldman, J. (2001). *Taking Your Talent to the Web: A Guide for the Transitioning Designer*. New Riders Publishing.

Zhou, C. (1987). Specifying communicating systems with temporal logic. *Temporal Logic in Specification*. Springer, Berlin. *Lecture Notes in Computer Science*, Vol. 398, pp. 304–323.

J. Besson is a researcher in computer science working at the Institute of Mathematics and Informatics, University of Vilnius, Lithuania. His main research interests include Data Mining, bio-informatics and web services and components composition. He got a Master degree in computer science from the Universite Joseph Fourier, Grenoble, France, and a PhD degree in Data Mining from INSA Lyon, France.

A. Lupeikienė is a senior researcher in the Software Engineering Department at the Institute of Mathematics and Informatics, Vilnius University, Lithuania. She also teaches as a part-time associate professor at the Faculty of Mathematics and Informatics. Her research interests include information system engineering, service-oriented system engineering, software engineering.

V. Medvedev was born in 1979 in Vilnius. In 2008 he received the doctoral degree in computer science (PhD) from Institute of Mathematics and Informatics jointly with Vilnius Gediminas Technical University. Recent employment is at the System Analysis Department of the Vilnius University, Institute of Mathematics and Informatics as researcher. His research interests include artificial intelligence, visualization of multidimensional data, dimensionality reduction, neural networks, data mining and parallel computing.

Realaus ir prognozuoto sistemos naudojimo palyginimas: web portalo atvejis

Jérémy BESSON, Audronė LUPEIKIENĖ,Viktor MEDVEDEV

Didėjant programų sistemų, apimant ir web portalus, sudėtingumui vis sunkiau išvengiama atotrūkio tarp numatyto ir realaus sistemos naudojimo. Išspręsti šią problemą galima lyginant išankstines sistemos naudojimo prielaidas su tikrąja naudotojų elgsena ir tokiu būdu išsiaiškinti, kaip pakeisti sistemos portalą. Realus naudojimasis portalu nustatomas analizuojant sistemos naudojimo pėdsakus duomenų tyrybos algoritmais. Sprendžiant adekvataus portalo užtikrinimo problemą siūloma taikyti tą patį formalizmą web portalo modeliui ir prognozuojamam bei realiam portalo naudojimui aprašyti. Inspiruoti BioCham idėjų straipsnio autoriai siūlo tokiu formalizmu pasirinkti laiko logiką ir Kripke struktūrą.