DOI: http://dx.doi.org/10.26483/ijarcs.v8i7.4445

ISSN No. 0976-5697

Volume 8, No. 7, July – August 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

AN APPROACH FOR AUTOMATING DESIGN OF WEB, ITS METRICS & KNOWLEDGE-BASE

Meenakshi Sridhar
Department of Computer Science & Applications
M.D. University
Rohtak, India

Nasib Singh Gill
Department of Computer Science & Applications
M.D. University
Rohtak, India

Abstract: In view of the fact that web applications are significantly large, complex, highly dynamic & critical software systems, it is desirable that processes of web design and development be systematic & be automated as far as possible. The task of automating an engineering activity is quite complex because it involves two types of contrasting intelligences: (i) Human intelligence which is rooted in informal expression, judgmental evaluation, inductive reasoning and commonsense, and (ii) The machine intelligence which is essentially based on formal expression, formal rule-based evaluation & deductive reasoning etc. Human intelligence is required for understanding the problem domains & their environments, and then for using the understanding for designing & developing solutions in such a form that it is understood and executed by the contrasting intelligence, viz. the machine intelligence.

The proposed approach takes care of the facts of human capabilities being rooted in informality & of machine capabilities being purely formal. For this purpose, solutions conceived by human experts—which are generally expressed in some natural language—are, first of all, translated to semi-formal mathematical entities: recursive lists. These recursive lists then are easily translated to fully formal entities in some functional programming language like LISP or Haskell. In this communication, the approach is applied to structural aspects of web and is explained through sufficient exemplars. The dynamics of web will be discussed subsequently.

Keywords: formal methods in engineering; automating web engineering; web metrics; human factors in engineering

I. INTRODUCTION

Web engineering, which is a relatively new discipline [1-5], is concerned with development and evolution of web applications, which are effective, robust & flexible in the sense that these can evolve over time as per changes in respective environments & requirements. The engineering is multidisciplinary involving contributions from diverse areas including software engineering, hypermedia/hypertext engineering, human-computer interaction & user interface design. In view of these facts, web applications are significantly large, complex, and even critical software systems. Formalization of software/web engineering design—which is a pre-requisite step for automated design, but yet to be applied sufficiently in design—has been discussed in [6-11]. Systematic, automated web engineering [12]—which applies computation to its various design, development & other activities—is highly desirable, rather, essentially required for developing such systems, specially, in view of their dynamic nature: web applications need to modify & evolve almost continuously over time. Partially or fully automating of the engineering activities can significantly help in timely delivery & in enhancing quality and productivity.

A task is said to be automated if it can be executed by the machine without human intervention. For this purpose, not only the relevant data structures and programs are expressed in a formal programming language, but even the control—by which, among other matters, the sequencing of execution of various programs etc. is done—has to be expressed in a formal, machine-readable & machine-executable form in some programming language.

For automating the control, the designers need to have still deeper understanding of the following: (i) how the control is achieved through human intervention (ii) how the machine—with different type of intelligence than that of human beings—can be made to have, by itself, similar power of control, through some software. Though, the designers have to be aware of these aspects, even when the control is not required to be automated, yet, in such cases, the awareness may be intuitive & informal. However, when the control is to be automated, the intuitive knowledge has to be further analyzed, has to be explicated, and finally, has to be formally expressed so that the control capability can be embedded, in the form of some data structures & programs, in the machine. Even when the programs are not to be fully automated, it has been pointed out by many scholars including [13-18], that formal definitions of problem domain, its relevant environment & interfaces between them are highly desirable, because, informal descriptions lead to ambiguous, incomplete & even inconsistent definitions.

Though mentioned implicitly in the previous paragraph, yet it may be explicitly stated that the task of automating various engineering activities for developing any software/web product involves two types of contrasting intelligences:

- Human intelligence required for understanding the problem domains, their environments & interfaces between them, and also, for understanding usability aspects and many other functionalities from the perspective of automating these activities. Human intelligence is rooted in informal expression, judgmental evaluation, inductive reasoning, and commonsense as major component of its knowledge base; and

- The computer intelligence to finally execute the automated activities. The computer/ computational intelligence is essentially based on formal expression, formal rule-based evaluation, deductive reasoning and explicit formal knowledge base.

Thus, an essential task in automating web applications development is to translate the informal human ideas about various engineering activities to the formal machine-executable form of these activities. Hence, for proper and correct automation, it is essential that the process—of translating various informally understood human activities to formal machine executable form—is understood very clearly by involved human experts and, possibly, even by the non-specialists including the users.

The proposed approach for automating web applications development etc. is based on this principle of essential human understandability of even the process of translating human understanding –which is generally informally, imprecisely, vaguely, and incompletely expressed—of various activities so far executed only by human beings into machine executable processes. However, the goal is more easily stated than is actually realized. The main hurdles include the fact that our understanding of the activities performed by us is deeply imbedded in human common sense and in subconscious mind. Both common sense and mechanisms of the subconscious mind are not consciously well-understood by us [13, 19-21]. However, in order to automate any human activity or capability, first of all, it needs to be consciously well understood. Then, the conscious understanding needs to be formally expressed. In this regard, attempts at explication of the implicit capabilities play a significant role. The proposed approach takes special care of the need for the explication aspect. Another strategy proposed by Brooks [19] and used in the approach is to develop software by growing it organically, adding more and more functions to systems as they are run, used, and tested.

The proposed approach is more comprehensive than just being about automating various activities of web engineering. It makes provision for storing relevant information & knowledge about the environment—which may consist of, among other entities, the expected users, and even the respondents to, say, a polling survey—in which the web artifacts function; and also about the interface—which may include, among other entities, the team of designers & developers, the tools, techniques, frameworks etc. required/used in the process; various metrics; and even other entities like planners of questionnaire for, say, exit polls.

After brief mention of the current Related Literature in the next section, we outline the approach in Section 3. Exemplars explaining the approach are given in Section 4. Final section concludes the work reported in the communication.

II. RELATED LITERATURE

In respect of the approaches/ frameworks proposed and tools developed for automating some facets of web development or usage, only a small sample of the recent attempts is outlined [23-27].

An automated approach for improving web service interface is discussed in [23]. A novel semantic knowledge engine using automated knowledge extraction from World Wide Web is proposed in [24]. In [25] a framework is proposed which makes easier integration of heterogeneous web-based systems. The proposed framework facilitates automation of Web Tasks. Automating the evaluation of usability aspects of web applications is discussed in [26], whereas, automated testing of web applications is discussed in [27].

In the relevant literature, including [23-27] mentioned above, the principle of essential human understandability—of the process of by which proposed approaches/ frameworks are arrived at and by which tools are developed, so that these can be understood, not only by the experts involved in proposing/ developing these, but even the non-specialist users—is apparently not visible.

III. DETAILED OUTLINE OF THE PROPOSED APPROACH

The proposed approach has the following major aspects

(i) Through the approach, various activities of design of web, which are initially expressed in some (informal) natural language, are first translated to semi-formal/ mathematical entities in the form of recursive lists, as mentioned above. The recursive lists so obtained are almost directly translated to formal expressions of a functional programming language like Haskell or LISP. Translation of recursive lists to any logical language like PROLOG is also almost straight forward.

(ii) The proposed approach is top-down (not necessarily strictly), distributed & flexible (easily expandable & modifiable). The Top-down aspect of the proposed approach may be sketched as follows:

- The whole www may be considered as a two-element list: (Communicating system, Computing system)
- Which may be, LOGICALLY, further expanded as list of lists: ((Internet, …,), (client-system, server-system)).
- The client system may be the same as the whole of computing system attached to various components of communicating system.
- The server system may be that subsystem of computing system, each computer on which has server capability
- Computing system on server side may be further considered as additionally having web-sites, web-server capability/software
- A client-computer may be logically considered as the list (Computer, browser)

- A communication may be structurally considered as a mathematical relation or list: (browser, search engine, server)
- Next, each of the terms like browser, server and search engine is further defined as recursive lists as will be explained later.

(iii) Further, there is flexibility in the approach in the sense that automation process need not be strictly top-down. For example, we have already formally defined *'generations of web'* as a recursive list, and further, which is to be a part of formal definition of web-site, but, web-site is yet to be defined formally in the form of a recursive list . Later on, when the term 'web-site' is formally defined as a recursive list, at that time the definition of 'generations of web' may be inserted as an element in the recursive list defining web-site. Also, if we have already defined web-site formally, in which, up to a particular point of time, 'generations of web' was not considered as one of its required constituent/ attribute. But, suppose later, it is found that 'generations of web' is also essentially required as constituent/ attribute. The approach allows the required modification easily.

(iv) The first step toward automation of a task is to develop a formal model [13-18] of the problem domain and its environment. The task of formalization may be divided in two major parts: (i) Formally defining the *structure* of the problem domain of the task, of the interface of the domain with the environment, and even of some relevant parts of the environment of the domain; and (ii) Formally defining the *dynamics* within the problem domain, and also dynamics of the interface of the problem domain & some relevant aspects of its environment.

The *structure* includes various elements and other components of the domain, relations and even meta-relations between these components. The *dynamics* involves various processes that go inside the domain (or interface) which either transform one state to another state of the problem domain while maintaining equilibrium in the domain, or transform the very structure of the domain/interface. We consider various issues in formalization of an engineering task/ process including its structure & dynamics.

A broad outline of iterative method for formalizing an engineering task may be expressed as follows:

For the structure

Step 1: Start with some central concepts of the relevant domain (and, sometime later, of the part of environment expected to interact with the problem domain). If two or more central concepts are considered, then find properties, if any of the concepts, relations & meta-relations etc. between these concepts. Express the properties of the considered concepts, relations & meta-relations between the concepts in terms of *mathematical* sets, relations and functions.

Step 2: Enumerate some (not all, not more than some pre-assumed number say 5) new closely related concepts to the central concepts, only after evaluating/ estimating the degree of closeness/relevance.

Step 3: Treat some of the concepts enumerated at Step 2 as central concepts, Go to Step1.

For the *Dynamics* of engineering process, in Steps 1, 2 & 3 above, replace the term 'concept' by the term 'process'.

In this respect, a number of formalisms have been proposed in literature [13-18]. A formalism using mathematical concepts of set, relation and function has been used among others by [13-15]. The proposed method of formalization has, at least the following advantages:

- Any functional programming language, say LISP or Haskell, or a logical language, say PROLOG, may be quite suitable for implementing the above delineated iterative process.
- The definition of a term, once defined, may be easily expanded/ modified later.
- A conceptual framework may be expanded easily both upwards as well as downwards. For example, a web browser displays the contents of a web page on a monitor etc. Further, if required, a monitor may be formally defined, in terms of density of pixels etc.

Next, the ideas explained above are employed in respect of automating the design of web, of relevant metrics and its knowledge-base.

A World Wide Web (WWW) model, as per Tim Berners-Lee et al [22], involves merging of the three techniques of hypertext, information retrieval, and wide area networking.

Internet, which provides the wide area networking for the purpose, is a sort of global rail-road-airways infrastructure along which any information can travel. It is in itself quite a vast & complex discipline & is beyond the scope of any design of WWW. Thus we discuss the other two aspects of automating of designing WWW.

WWW constitutes a global information space consisting of billions of web pages, with a web page being a unit of WWW. A web page is mainly a text document, just big enough so that it can be displayed, one at a time, on the display terminal of a computer or mobile device. In addition to text, a web page may also contain images, video, audio, and software components. The text in the web page may be formatted & annotated with Hypertext Markup Language (HTML). Also, a web page incorporates hypertexts, text with hyper-linking capability, each hypertext allowing accessing of related information scattered across more than one pages.

At the next higher level of top-down structure of WWW is website, where, multiple web pages with a common theme, a common domain name, or both, make up a website. As per the iterative process delineated above, the process of formalization is initiated with formal definitions of the two central concepts of 'web-site' & 'web page'.

IV. EXEMPLARS EXPLAINING THE PROPOSED APPROACH

From the point of view of developing a robust system, which may remain useful over a relatively long period, we need to have a unified & integrated view of the system under consideration which is well situated in its environment. From this perspective, concerns about the system, from the top level, may be considered/ classified as

- Domain/ System concerns
- Environmental concerns
- Interface concerns

Each of these concerns may be further considered as composed of

  - Technical concerns
  - Conceptual concerns

Some of the environmental concerns may include social concerns, e.g. social impact of a new technological system.

Next, some exemplars for these types will be considered

A website

- is identified with/by a common domain name,
- is published on list of (at least one) web servers,
- is dedicated to a particular topic or purpose, e.g. entertainment, social networking, providing news, and education,
- contains content *regarding* an individual, an academic institution, some business, government or other profit/non-profit organization,
- is accessible via a public Internet Protocol (IP) network, such as the Internet, or a private local area network (LAN), by referencing a uniform resource locator (URL) that identifies the site.
- is constituted of web pages, each web page is basic component/ unit of web site, that can be displayed, one at a time, on the display terminal of a computer or mobile device.

Web-site may be expressed semi-formally in a LISP-like language as

web-site (

Working-definition *(*"a set of related web pages located under a single domain name"),

identifier-cum-types (static, dynamic, Document centric, Transactional, workflow-based, collaborative, Social Web Applications),

components/ structure (**is-identified-by** (domain-name), **is-constituted-of** (list-of web-pages), **is-published-on** (list of web servers), **contains-content-regarding** (name of individual/ academic-institution/business/ government…), **accessible-via** (a public Internet Protocol network (Internet), private local area network (LAN)), **accessible-through** (URL), **purpose** (Entertainment, Social networking, Education, Providing news …),

features/ characteristics (appearance (colour, text, meaningful graphics, quality photography, simplicity),**Usability (**simplicity, fast loading pages, minimal scroll, consistent layout, logical navigation, descriptive link text, screen resolution), **generations (web1.0** (year (1989), contents(static), communication(producer→consumer), protocols (HTML, HTTP…), examples(Britannica Online, Home Pages)), **web 2.0** (year (2003), content (Dynamic), communication (producer←→consumer), technologies-and-services(wiki, blogs, server-side scripting, Javascript, AJAX) protocols (HTTP1.1, SOAP, XML,RSS), …), examples(Wikipedia, blogs/wikis)), **web 3.0** (year (future), contents and services (ubiquitous, pervasive), communication(producer ←→ consumer←→machine), technologies-and-services(Intelligent-Agents, Semantic web, smart interfaces, intelligent search, 3d web) protocols (HTTP1.1, SOAP, RDF,RDFS, OWL, SPARQL, Open APIs), examples(semantic web, lifestreams)),

internal-processing/ functioning-of-website (…)),

where, in the ordered pair (x (y)) on different lines above 'x' denotes some attribute-name of web-site and '(y)' denotes corresponding list of attribute-values.

The above semi-formal definition of web-site may be recursively obtained/ constructed as follows:

Step1 Initial Step: At the top a web-site may be considered as a list

web-site (Working-definition, identifier-cum-types, components/ structure, features/ characteristics, generations, internal-processing/functioning-of-website)

The entity 'web-site', at first, is defined in terms of its top level attributes, viz. Working-definition, identifier-cum-types etc. as a list

(Working-definition, identifier-cum-types, components/ structure, features/ characteristics, generations, internal-processing/functioning-of-website).

Step2 Recursion Step: Next, let some or all of the attributes need to be expanded. Let the attribute 'generations' of 'web-site' is defined semi-formally as a list as follows

generations (web 1.0, web 2.0, web3.0).

This expanded semi-formal definition of 'generations' replaces the attribute 'generations' in the definition of web-site to get

Web-site (Working-definition, identifier-cum-types, components/ structure, features/ characteristics, **generations (web 1.0, web 2.0, web3.0)**, internal-processing/functioning-of-website).

The Recursion Step may now be applied to each of web 1.0, web 2.0, web 3.0. For example, web 1.0 is further defined as the list

web1.0 (year, contents, communication, protocols, examples)

The expanded definition of web-site then becomes

Web-site (Working-definition, identifier-cum-types, components/ structure, features/ characteristics, generations **(web1.0 (year, contents, communication, protocols, examples),** web 2.0, web3.0), internal-processing/functioning-of-website).

The salient feature of the approach, illustrated above, is that it is easily understood by human-experts as well as others and, at the same time, can be further expressed, without difficulty, in machine-executable form.

Similarly, the other elements/ components of web are defined using the approach. Next, formalization of another fundamental concept of the field of web design, viz. 'web page' is discussed.

Web-page may be expressed semi-formally in a LISP-like language as

web-page(

working-definition ("a hypertext document connected to the World Wide Web"),

identifier-cum-types (static, **Dynamic** (server side web page, client side web page)),

components/ structure (**type-of-info** (**textual**, **non-textual** (**audio** (MP3, Ogg, …), **static-images** (GIF, JPEG, PNG, SVG,…), **animated-images** (animated-GIF, SVG,…), **video** (WMV, RM, FLV, …)), **interactive-info** (**on-page** (buttons, interactive text, interactive illustrations), **between-page** (hyperlinks, forms)), **accessed-through** (protocol (HTTP, HTTPS)), **accessed-via (**browser),**internal-info** (**comments**, **linked-files-via-hyperlink** (DOC, XLS, PDF, …), **metadata** (semantic meta-information, Charset information, Document Type Definition), **diagrammatic-and-style-information (CSS), scripts** (javascript, functionality, complement interactivity)), **displayed-on** (computer, mobile)),

features/characteristics (appearance (colour, text, meaningful graphics, quality photography, simplicity), **Usability (**simplicity, fast loading pages, minimal scroll, consistent layout, logical navigation, descriptive link text, screen resolution) **),**

internal-processing/functioning-of-webpage (…))

Another important component Web-browser of World Wide Web may be expressed semi-formally in a LISP-like language as

web-browser (

working-definition ("A **web browser** (commonly referred to as a **browser**) is a software application for retrieving, presenting and traversing information resources on the World Wide **Web**. An information is identified by a Uniform Resource Identifier (URI/URL) that may be a **web** page, image, video or other piece of content."),

identifier-cum-types (crome, foxpro,…),

components/structure (user-interface (menu (print, save a favorite webpage, set internet options, …), **function icons** (**buttons** (print, refresh web page, go back to the last page viewed, go forward to the next page viewed,…)), **buttons** (minimize, maximize, close), **web address area**, **status bar**, **viewport**, **layout-engine, rendering engine**, **JavaScript-interpreter, UI-backend, networking-component, data-persistence-component**

features/characteristics (speed (fast to launch, fast to run applications, fast to search and navigate, fast to load webpage), **simplicity** (easy to search and navigate, efficient tab management, tab to search, built-in PDF viewer, start from where you left off), **security (**safe browsing, sandboxing, auto-updates), **privacy (**incognito tab, privacy preferences, clearing browsing data, additional privacy resources), **customization (**adding new users, extensions, themes)),

internal-processing/ functioning-of-webpage (…))

Another important component Search engine of World Wide Web may be expressed semi-formally in a LISP-like language as

search-engine(

working-definition ("Search Engine is a program that searches for and identifies items in a database that correspond to keywords or characters specified by the user, used especially for finding particular sites on the World Wide Web"),

identifiers-cum-types (crawler-based search engines (google, alltheweb, altavista), **Human-powered-directories (**yahoo directory, looksmart, open directory)),

components/**structure** (**webcrawler** (included-pages, excluded-pages, **document-types** (HTML, DOC, PDF, EXL,…), frequency-of-crawling), **database (**size-of-database, freshness-of-database), **search- algorithm (**operator (AND, OR, NOT), phrase-searching, truncation), **ranking-algorithm** (location-and-frequency, link-analysis, clickthrough-analysis)),

features/characteristics (Advanced site search engine, Reports track visitors' searches, Automatic site map, Automatic what's new list, Optional web search, Complete customization, Scheduled re-indexing, Content monitoring, No fixed page limit, Ease of use, Indexing of password-protected pages),

internal-processing/functioning-of-search-engine (document-processor (Normalizes the document stream to a predefined format, Breaks the document stream into desired retrievable units, Isolates and metatags subdocument pieces, Identifies potential indexable elements in documents, Deletes stop words, Stems terms, Extracts index entries, Computes weights, Creates and updates the main inverted file against which the search engine searches in order to match queries to documents),**Query-processor** (Tokenizing, parsing**,** Stop list and stemming, Creating the query, Query expansion, Query term weighing), **Search-and-matching-function)**)

Another important component Web-server of World Wide Web may be expressed semi-formally in a LISP-like language as

web-server(

working-definition ("A **Web server** is a program that uses HTTP (Hypertext Transfer Protocol) to serve the files that form **Web** pages to users, in response to their requests, which are forwarded by their computers' HTTP clients. Dedicated computers and appliances may be referred to as **Web servers** as well.")

identifiers-cum-types (shared , dedicated, Multi-process (Apache on Unix) „ Multi-threaded (Apache on NT/XP) , Single process event driven (Zeus, thttpd) , Asymmetric multi-process event-driven (Flash))

components/structures (data-analysis (who is visiting a website, how long viewed the site, the date and time of each visit, which pages were displayed, **Physical capacity of the server** (computing power, storage and memory)), **web-log-file-analyser-program (**analog web server log file analyzer program, WebTrends web server log file analyzer program), **link-checkers (**Elsop Linkscan, Big Brother software), load limiters, **database (**MySql), **architecture(** two-tier client server architecture, three tier client server architecture, n tier client server rchitecture), **operating system (**Windows NT server, Windows 2000 advanced server, Microsoft. NET), **web server software (**Apache, HTTP server, Microsoft Internet Information System, …), **Network and/or Internet connectivity** (modes of connection and the number of concurrent users it can support), , **Performance and quality of service** (latency, throughput, low memory utilization), **Application tiers** (type of different applications deployed on the server), **Platform supported** (.Net, LAMP)),

features/characteristics (document root, server root, virtual document tree, virtual host , proxy server, logging , Security (access control) , Traffic analysis),

internal-processing/functioning-of-search-engine (Browser Resolves the Domain Name to an IP Address,

Browser Requests the Full URL, Web Server sends the requested Page, Browser Displays the Webpage))

Next, we consider some exemplars related to web metrics [28-30].

Web metrics (

compatibility-metric (

working-definition (" This metric comprehensively test a website, and check if it successfully displays across various browsers, platforms and resolution"),

type (desktop-browser-compatibility (Chrome, Firefox, Safari, Opera, …), **mobile-browser-compatibilty** (iPhone, iPad, Android, Blackberry)),

components (HTML tags, CSS, Image formats, Technologies (Flash, Max-Q tech, …),

formal-definition ())

performance-metric (

working-definition (" This metric refers to the **speed** in which **web** pages are downloaded and displayed on the user's **web** browser"),

type (Server-side-metrics (Total Page Views per Week , Total Hits per Week , Total User Sessions per Week , Average Page Size , Average Hit Size , Page Request Distribution, User Abandonment), client**-side-metrics (**interaction speed, latency tolerance, familiarity, connection speed)),

components (),

formal-definition (),)

usability-metric (

working-definition (" This metric checks the ease of use of a website." **),**

components (success rate, the time a task require, the error rate, users-subjective satisfaction),

formal-definition (),)

Accessibility-metric (

working-definition (" This metric checks **whether a websites** is accessible by people with disabilities."),

types (sensory-metric, motor-metric, cognitive-metric),

formal-definition (),

) **)**

V. CONCLUSION

One of the significant hurdles in automating an engineering task is about how to translate informal ideas of human experts about the proposed solutions of engineering tasks to machine executable solutions. In this communication, a systematic approach is proposed that first translates the informal human solutions to semi-formal mathematical entities, viz. recursive lists. Then, it is explained how these recursive lists are translated to formal expressions in a functional programming language like LISP or Haskell.

In view of the fact that there is large number of programmed artifacts required for complete automation, here, only some exemplars are discussed and these deal with only the structural aspects of web design. The dynamic aspects of web design will be discussed in some subsequent paper.

VI. REFERENCES

[1] Casteleyn S., Daniel F., Dolog P. & Matera M.(2009) Engineering Web Applications, Springer

[2] Gerti Kappel, Birgit Proll, Siegfried Reich (editors) (2006) Web Engineering: The Discipline of Systematic Development of Web Applications John Wiley

[3] Gustavo Rossi et al (editors) (2008) Web Engineering: Modelling and Implementing Web Applications Springer Verlag

[4] Emila Mendes, Nile Mosley (editors) (2006) Web Engineering Springer Verlag, San Murugesan & Yogesh Deshpande (editors) (2001) Web Engineering LNCS 2016, Springer Verlag

[5] Formal Methods in Software Engineering by Stephan Schulz, http://www4.in.tum.de/~schulz/TEACHING/CS63Z-2005.html

[6] John Cooke (2005) Constructing Correct Software Second Edition Springer

[7] D. Bjørner (2006) Software Engineering 3 Domains, Requirements, and Software Design springer

[8] Almeida et al (2011) Rigorous Software Development: An Introduction to Program Verification, Springer

[9] I.H.M. van (2009) Applying Formal Specifications in Web Design – A Comparative Study Fourth International Multi-Conference on Computing in the Global Information Technology, Coppenhagen

[10] Donald Sannella, Andrzej Tarlecki, (2012) Foundations of Algebraic Specification and Formal Software Development Springer

[11] Automated Software Engineering, An International Journal, ISSN: 0928-8910 (Print) 1573-7535 (Online)

[12] Meenakshi Sridhar, Naseeb Singh Gill (2015) Imperfection of Domain Knowledge and Its

[13] Formalization in Context of Design of Robust Software Systems, Journal of Software Engineering and Applications, 2015, 8, 489-498

[14] Meenakshi Sridhar, Naseeb Singh Gill (2015) Formal Conceptual Framework for Structure of Context of Component-Based System for Designing Robust Software Systems & Metrics, International Journal of Computer Applications (0975 – 8887) February 2015

[15] C. Serban, A. Vescan and H. Pop, (2010) A Conceptual Framework for Component-based System Metric Definition

[16] M. Goulão, F. Abreu,(2005) *Formalizing metrics for COTS*, Department of Informatics, Faculty of Sciences and Technology, New University of Lisbon,2825-114 Monte de Caparica, Portugal, 2005.

[17] J. Woodcock, P. Larsen, J. Bicarregui and J. Fitzgerald, (2009) *Formal Methods: Practice and Experience*, ACM Computing Surveys, Vol. 41, No. 4, Article 19, Publication date: October 2009.

[18] J. Zhixiong, L. Qian, and X. Pen,(2007) *A Formal framework for description of semantic web services* , Seventh International Conference on Computer and Information Technology 2007 IEEE, DOI 10.1109 10.1109/CIT.2007.24.

[19] Brooks, F.P. (1986) No Silver Bullet—Essence and Accident in Software Engineering. *Proceedings of the IFIP Tenth*

[20] Mair, C. and Shepperd, M. (2011) Human Judgement and Software Metrics: Vision for the Future ICSE'11. 21-28 May 2011, Honolulu.

[21] Harman, M. (2012) The Role of Artificial Intelligence in Software Engineering RAISE 2012. Zurich.

[22] Berners-Lee T., Cailliau R., Groff J-F, and Pollermann B., (1992) World-Wide Web: The Information Universe, pp. 52-58 Vol.2/No. 1 Electronic Networking ■ Spring 1992

[23] Ali Ouni, Zouhour Salem, Katsuro Inou, Makram Soui, *(2016)* SIM: An Automated Approach to Improve Web Service Interface Modularization, 2016 IEEE International Conference on Web Services

[24] Venkatesh Mabbu and Abu Asaduzzaman, Muhammad F. Mridha (2016) A Novel Semantic Knowledge Engine Using Automated Knowledge Extraction from World Wide Web, 5th International Conference on Informatics, Electronics and Vision (ICIEV)

[25] Chun-Hsiung Tseng, Yung-Hui Chen, Yan-Ru Jiang,Pin-Yu Su,Fang-Chi Tsai

[26] (2016) Automating Web Tasks by Simulating Browser Behaviors, 2016 International Conference on Platform Technology and Service (PlatCon), 15-17 Feb. 2016

[27] Nouzha Harrati, Imed Bouchrika, Abdelkamel Tari and Ammar Ladjailia (2015) Automating the

[28] Evaluation of Usability Remotely for Web Applications via a Model-Based Approach, 2015 First International Conference on New Technologies of Information and Communication (NTIC), 8-9 Nov. 2015

[29] Riihimäki T. (2014) Evaluating the Value of Web Metrics, Thesis submitted by, Department of Information and Service Economy, School of Business, Aalto University,.

[30] C l i f t o n B. (2012) Advanced Web Metrics with Google Analytics Third Edition by, John Wiley

[31] Dhyani D., Ng W. K, Bhowmick S. S., (2002) A Survey of Web Metrics, ACM Computing Surveys, Vol. 34, No. 4, December 2002, pp. 469–503