SCIENTIFIC PAPERS, UNIVERSITY OF LATVIA, 2009. Vol. 751
COMPUTER SCIENCE AND INFORMATION TECHNOLOGIES 7–21 P.

Web Site Modeling and Prototyping Based on a Domain-Specific Language

Agnis Stibe, Janis Bicevskis
University of Latvia, 19 Raiņa Blvd, LV-1459, Rīga, Latvia
Agnis.Stibe@gmail.com, Janis.Bicevskis@lu.lv

In the history of software development, there is a haven of different methodologies, approaches, and tools that have been designed to capture the expectations of customers and transfer them into requirements for information systems understandable by programmers. However, reality shows that the percentage of failures in development of information systems in terms of time, money, and functionality is not decreasing.
This paper describes creation of a principle for and development of a domain-specific web site modeling language in order to make interactive dialogue during the very beginning of web site development process to match the expectations of both the customer and supplier. We have developed a tool to implement all the possibilities offered by language. The web site modeling tool has a corresponding component to each general function of the language. The web site model can be simulated within the tool and serve as a prototype for the desired web site in reality. Prototyping approach eliminates barriers between business and information technology people, leading to common understanding of web site goals and success in delivery and implementation. For a more practical view on the issue, the example of the State Revenue Service of the Republic of Latvia web site is examined in this paper.

Keywords: software engineering, modeling, specification languages, domain-specific languages, web requirements.

1 Introduction: the Variety of Requirements' Specifications

One of the most marked problems in development of any kind of information systems (IS), including web site development, is gathering and consolidation of adequate requirements. Professionally collected, they determine not only the functionality of the developed web site, but also serve as requirements for testing at the phase of acceptance. The requirements' analysis is considered to be a key step in the development of successful IS by all software engineering approaches [1]. Empirical data demonstrate that efforts invested in a detailed requirement analysis considerably reduce drawbacks in later phases of the development [2].

One of the following approaches is most frequently applied in development of requirements.

- Requirements are formulated by business-oriented people, who are less experienced in information technology (IT) issues. As a result, requirements are more informal, incomplete, and sometimes even inconsistent, leaving them to the imagination and interpretation of IT specialists and programmers. That leads to many unexpected changes right after the customer starts to use the newly developed web site.
- Requirements are formulated by IT specialists, which commonly have poorer knowledge of the web site's role and business processes. As a result, there are very well defined requirements that are understandable to web site developers, but may appear to be contradictory to the needs the web site is being produced for.

In practice the solution is quite often based upon the attempts to unite both business-oriented people and IT specialists into a team. Then again, the important question is the choice of the requirement specification language. If requirements are written in a formalized language with explicit semantics, later execution of that specification certainly helps to escape misinterpretation of requirements provided earlier. Unfortunately, Unified Modeling Language (UML) [3] that is currently often recommended is understandable to IT specialists but is less acceptable for business-oriented people. Therefore, UML usage in practice is limited and specifications quite often are written in a natural language, which, in turn, causes ambiguity, inaccuracy, and disagreement about consistency between the specification and the developed system. Common understanding of the planned system – unified communication language – is especially important at the very beginning of system development, when requirements and proposed solutions need to be understood equally by both sides. In many cases the solution is using combined specialized Business Process Modeling Languages (BPML) [4] and workflows. Nevertheless, for the development of web sites (portals, home pages, etc), business modeling languages are less appropriate.

In this paper, a simple and practical reasoning-based web site specification language is proposed. It could, at least partially, solve communication problems between business oriented people and IT specialists.

2 Statistics: an Example of Problem Segments in Web Site Development

Below we present an example which will highlight the main ideas of web site specification. A project of national importance – the project of development and maintenance of the portal of the State Revenue Service (SRS) of the Republic of Latvia [5] – is chosen as an example. It is characterized by:

- quality specification, because the developer of the portal was selected by public procurement, where the procurement subject must be clearly defined. Additionally, there was a previous version of the portal in operation, and that helped to formulate requirements for the new portal more precisely;
- a competent developer team with previous experience in development of several portals of national importance.

The statistics about the development and maintenance of the portal were collected from the first quarter of 2005 to the third quarter of year 2008. The acceptance testing of the portal was performed by the customer during first and second quarter of year 2005. The portal was launched and maintenance began in the second quarter of 2005. All events during testing and maintenance of the portal, further "problem events", were recorded by the customer with the help of a computerized problem recording system. Consequently, the statistics represent only the customer's view on the project and do not include the developer's internal communication. It is remarkable that these statistics are complete, meaning that the developer only processed problem notifications officially recorded by the customer, and no other way of communication regarding drawbacks of the portal was accepted. That established reliable ground for completeness and credibility of the collected data.

234 problem notifications were recorded altogether. They can be divided into five general groups represented in Figure 1.

- Errors (55). This group includes only those problem notifications which are obviously interpretable as inconsistency between the operation of the portal and the specification.
- Misinterpreted requests (32). A programmer has developed an application that he or she thinks is consistent with the specification. However, the customer founds an inconsistency with the specification during the phase of acceptance-testing, and the programmer has to admit his or her fault.
- Loose requests (25). Problem notification is recorded, but it cannot be treated as a mistake of a programmer because the formulation in the specification is ambiguous.
- Changed requests (83). The customer changes initial requirements during acceptance-testing and maintenance of the portal, which results into changes within the application.
- New requests (39). The customer formulates new requirements that were not stated in the beginning.

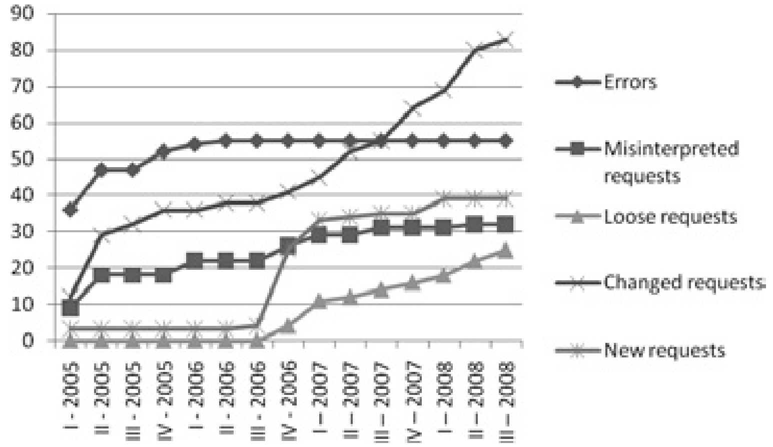


Fig. 1. Problem notifications (cumulative)

The statistics show that mistakes of programmers comprise only 23% of all problem notifications, which lets us question the common belief about the inevitability of programmers' faults. Especially surprising is dispersion of problem notifications in time, which shows that programming errors are solved relatively fast, but the amount of changed requests and additions during the operation of the portal remains stationary high. It is also supported by the fact that 52% of problem notifications correspond to changed requests and new requests; in addition, their amount does not decrease during the maintenance. As a result, the developed but at the same time informal specification was overwritten many times. Actually, there is no unified specification any more, except the one proposed in this paper. The experience of the SRS project demonstrates that similarly to workflow languages that offer specification options to describe operation of information systems, there is a necessity for a particular specification language for the development of web sites, portals, etc, which would let describe easily the main elements of the web site, retaining close relation with the web site operated in reality during its whole lifecycle.

The previously analyzed statistics on the basis of a particular example clearly highlight the following statement: in projects that include development and maintenance of the web site for several years, the main problem is not correction of programmers' mistakes but changed requests in the existing specifications and new requirements. Therefore, a solid ground is established for development of a domain-specific language (DSL) for description of web site elements and their functions.

3 The Research: Domain-Specific Language for Web Site Modeling

In all branches of science and engineering, generic and specific approaches can be distinguished. A generic approach provides a general solution for many problems in a certain area. A specific approach provides a better solution for a smaller set of problems. Both approaches are considered in computer science in relation to the topic "domain-specific languages versus generic programming languages" [6]. A domain-specific language indicates a specification language dedicated to a particular problem domain – web site development in this paper.

On the one hand, currently more efforts are aimed at evolving the existing Semantic Web [7] concepts defined by the World Wide Web Consortium [8] (Resource Description Framework – RDF [9] and Web Ontology Language – OWL [10]) and Universal Networking Language – UNL [11] specifications. Compatible with these concepts is Web Site Parse Template [12], which is a specification for web site structure and content description for web crawlers. It is an effective way to provide web crawlers with proper web page templates to parse web site content more accurately, coordinating the same object attributes used in different pages of the same web site. The Web Ontology Language (OWL) is designed for use by applications that need to process the content of information instead of just presenting information to humans.

For many years, researchers put their efforts in the Semantic Web Service area, work toward further standardization in the area of Semantic Web Service languages and a common architecture and platform for Semantic Web Services [13]. The Web Service Modeling Ontology (WSMO) [14] provides a conceptual framework and a formal language for semantic description of all relevant aspects of web services in order to facilitate the automation of discovering, combining, and executing electronic services over the Web. Web Service Modeling eXecution environment (WSMX) [15] is the reference implementation of WSMO. It is an execution environment for business application integration where enhanced web services are integrated for various business applications. WSMX internal language is Web Service Modeling Language (WSML) [16]. WSML is based on different logical formalisms, namely Description Logics, First-Order Logic, and Logic Programming, which are useful for the modeling of Semantic Web services. Semantic Web is more perceived as a vision for the future of the Web, in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the Web.

Another approach is Web Modeling Language (WebML) [17], a notation for specifying complex web sites at a conceptual level. WebML enables high-level description of a web site by distinct orthogonal dimensions: its data content (structural model), the pages that compose it (composition model), the topology of links between pages (navigation model), the layout and graphic requirements for page rendering (presentation model), and the customization features for one-to-one content delivery (personalization model). All the concepts of WebML are associated with a graphic notation and a textual XML syntax. WebML specifications are independent of both the client-side language used for delivering the application to users, and of the server-side platform used to bind data to pages, but they can be effectively used to produce a site implementation in a specific technological setting. WebML guarantees a model-driven approach to web site development.

Significant characteristics that show why WebML is not suitable for the aim of this paper is that WebML: is a high-level specification language for designing data-intensive web applications; stresses the definition of orthogonal navigation and composition primitives, which the designer can arbitrarily compose to model complex requirements; includes an explicit notion of site view, whereby the same information can be structured in different ways to meet the interests of different user groups or to obtain a granularity optimized for users approaching the site with different access devices; covers advanced aspects of web site modeling, including presentation, user modeling, and personalization.

The UML-based Web Engineering (UWE) [18] approach provides a set of web domain-specific model elements for modeling different concerns describing a web system, such as content, hypertext structure, presentation, and processes. These model elements and the relationships between them are specified by a metamodel. UWE's notation is defined as a lightweight extension of the UML providing the so-called UML Profile for the Web domain. The main focus of the UWE approach is to provide a UML-based domain-specific modeling language, model-driven methodology, tool support for systematic design, and tool support for (semi-)automatic generation of Web applications.

On the other hand, approaches mentioned before are far too sophisticated when the need is to gather, store, and validate customers' requirements for a web site. The solution for any particular problem is going to be discovered and explored in the phase of requirement specification during the process of deeper acquaintance between the customer and supplier. A customer has to be able to define the web site's processes in a simple and precise manner (Web Site Model), at the same time not going deep into programming details.

In turn, a supplier a designs web site prototype and demonstrates it to the customer, who has a chance to evaluate and assess the supplier's proposed solution and its compliance to initial requirements at the very beginning of the web site development.

The approach presented in this paper aims to build an effective environment for collecting, storing, and modeling requirements for web site development, and to make a prototype web site based on these requirements afterwards, allowing everyone to ascertain that the result will meet the expectations. The development of a domain-specific language typically involves a sequence of steps starting with identification of the problem domain, gathering of all relevant knowledge in this domain, and ending with design and implementation of a compiler that translates DSL programs.

In many cases DSL serves as a basis for the development of operable prototypes, which in turn are very good means of mutually harmonized communication for requirement gathering and specification of functionality of developed systems. A deeper insight into prototyping is considered to be a natural extension of the research work, but this paper focuses on the definition of a domain-specific language designed for modeling of web sites, and does not aim to describe the syntax and semantics of the language at this stage.

4 Contribution: Web Site Modeling Language (WeSiMoLa)

Reviewing the history of recent tendencies in requirement specification, two most common ways can be singled out: one way is high-level specification, where general requirements are specified with no intention to go into the smallest details and functions; the aim is to prepare specification as far as it is understandable for developers in terms of the expected result. The other way is deep programming, where specification includes every detail of the planned system as far as describing each smallest action and reaction to it. In this paper, the first alternative is chosen, because web site environment is developing and changing so fast that it is more reasonable to make high level specification in much shorter time.

The following solution is presented to the problem described in this paper:

- firstly, a web site modeling language, called WeSiMoLa, is created. Additionally, all terms that are simple enough but at the same time satisfactory to formulate web site requirement definition are defined;
- secondly, realization (interpretation) of WeSiMoLa is offered in the form of a prototype, which can be demonstrated to the customer as the web site's prototype.

There are several advantages to the proposed approach. It allows to create an efficient dialog between the customer and supplier in the early stage of requirement specification, enabling to escape the risks of wasted resources (time, money, or others) during elimination of misinterpreted expectations set in inaccurate specifications. The approach assures that the customer always will have specification of their web site that is in compliance with the real up-and-running web site.

WeSiMoLa is a domain-specific language aimed to describe web site's layouts, content, modules, and information structure in an easy readable and constructible manner. WeSiMoLa allows to record web site's content and navigation net in an understandable form that is convenient for other potential users.

The general concept of WeSiMoLa is built upon the idea of the FrameSet approach [19]. The fundamental elements of the language are simple Objects common on the Web, like text, picture, link, etc. As the next step, those objects are combined in different ways to make Frames. Each Frame is build with a specific purpose, e.g., archive, registration forms, questionnaires, and related topics. Finishing the concept of WeSiMoLa, Frames are going to be allocated in FrameSets. The structure can be defined for each FrameSet in terms of number of rows and columns which divide the screen or general view of the web site into several parts. Each of these parts can be filled with one or more Frames.

4.1 FrameSet Definition

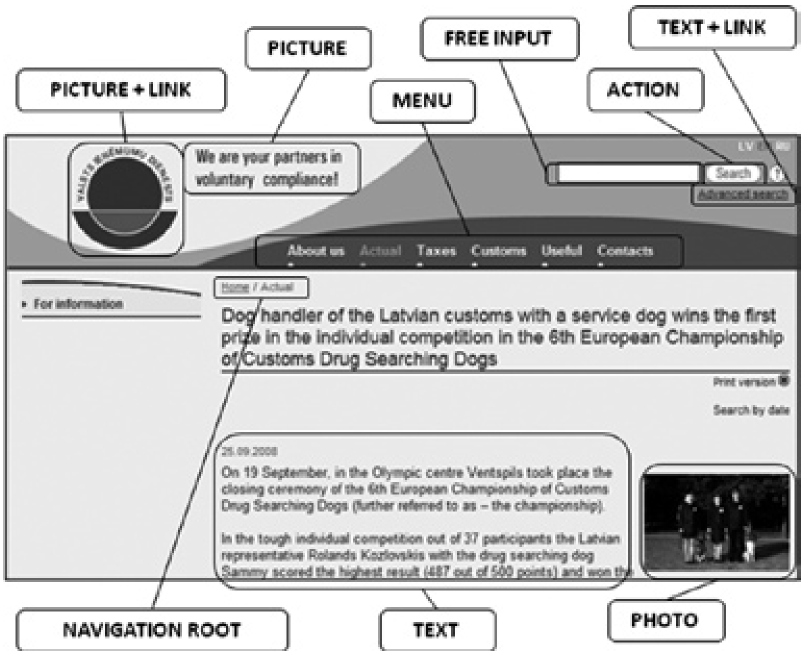
FrameSet is a structure that divides the total space into number of rows and columns, meaning everything that can be viewed by a web browser (window), including space that is outside the screen, which can be reached by scrolling. Rows and columns mark the areas that will serve for a place where Frames will be allocated. It is a physical and logical division that later is visible on the screen. Visibility is enhanced by graphical design. If not, there are always invisible logical lines that separate groups of Frames on the screen. Each area in the FrameSet, whether it is row or column, has a fixed or proportional height and width. In case of fixed parameters, a row or column will remain at a constant size when browser window is resized or resolution is changed. If height or width of some row or column is defined as proportional, the size of the particular area will change according to the updated resolution or proportionally to the resizing of the browser window. Any area in the FrameSet can hold none, one, or many Frames.

4.2 Frame Definition

Frame can be allocated somewhere in a FrameSet according to its purpose or size. Similarly to FrameSet areas, there can be two kinds of Frames – with fixed height or width or of flexible size that adapts to the size values of an area where it is placed. Each Frame has its purpose or shape. A Frame with a purpose means that it is built to execute a particular process or to serve a particular need. Most common types of purpose Frames are help, archive, registration forms, questionnaires, etc. All other Frames are defined with an aim to represent particular information in a particular shape, e.g. a picture and a textual description, which are Objects. Frames are built with the help of various Objects, combining them to serve the particular purpose of the chosen Frame. Frame has a parameter that allows to show it in a pop-up window or as an overlaying window as well.

4.3 Object Definition

Objects are the basic elements of WeSiMoLa. They are different and each of them performs a specific task or represents a unique meaning. There are passive Objects like text and graphical elements without the possibility to interact with them; they are only representative elements. All others are active Objects like links in a text or graphical elements; it is possible to navigate away from it, input data and perform a related process with the help of various input fields and action buttons, etc. The number and type of Objects is predefined in the language. The only task is to develop Frames from the Objects and put the Frames into FrameSets.



We are your partners in voluntary compliance!

Dog handler of the Latvian customs with a service dog wins the first prize in the individual competition in the 6th European Championship of Customs Drug Searching Dogs

26.09.2008

On 19 September, in the Olympic centre Ventspils took place the closing ceremony of the 6th European Championship of Customs Drug Searching Dogs (further referred to as – the championship).

In the tough individual competition out of 37 participants the Latvian representative Rolands Kozlovskis with the drug searching dog Sammy scored the highest result (487 out of 500 points) and won the

Fig. 2. WeSiMoLa Objects in SRS web site

List of predefined Objects:

- Text Element;
- List;
- Table;
- Picture;
- Animation;
- Interaction;
- Link:
  - Internal Frame / Internal Web Site / External,
  - To Page / to File;
- Input Fields:
  - Radio Buttons,
  - Check Boxes,
  - Drop Down,
  - Free Input;
- Action on Input Data;
- Menu:
  - Horizontal / vertical,
  - Levels overlapping horizontally / vertically,
  - Levels expanding vertically,
  - Language;

- Photo / Audio / Video;
- Search;
- Navigation Root;
- Banner;
- Blog;
- Calendar;
- Abstract;
- Login;
- Print this page.

An example of part of WeSiMoLa Objects is represented in Figure 2. For consistency throughout the paper, SRS web site is taken as the example.

5 The Web Site Modeling Technique

The basis of the web site modeling technique is composing Frames using predefined Objects, allocating Frames into designed FrameSet templates, and building up the navigation. To support this modeling technique, a Web Site Modeling Tool (WeSiMoTo) is developed. Four general components of web site modeling are:

- information source structured in a tree form for menu purposes and core navigation:
  - in WeSiMoTo this component is embodied in Tree Builder;
- repository for keeping composed Frames:
  - in WeSiMoTo this component is embodied in Frame Composer;
- repository for keeping designed FrameSet layouts:
  - in WeSiMoTo this component is embodied in FrameSet Designer;
- navigation modeling:
  - in WeSiMoTo this component is embodied in Navigation Net.

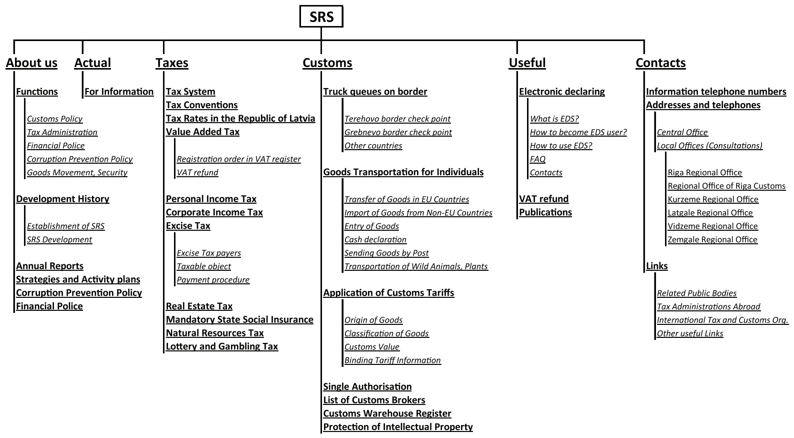


Fig. 3. Full Information Tree of the SRS web site

5.1 Information Tree Modeling (Tree Builder)

The information tree represents graphically all the textual information that a web site has and its location in a structured way. It also clearly shows the core navigation through the web site and subordination of information pieces.

Full Information Tree of the English version of SRS web site is represented in Figure 3. It has six general sections and goes deep to the fourth sublevel.

Tree Builder is the component in WeSiMoTo that allows to input and order information in an easy understandable and user friendly environment. Its general functions are:

- create a new tree / save / remove;
- add / remove language of the tree;
- add / edit / remove sublevels of information in the tree;
- add / edit / remove information;
- save tree.

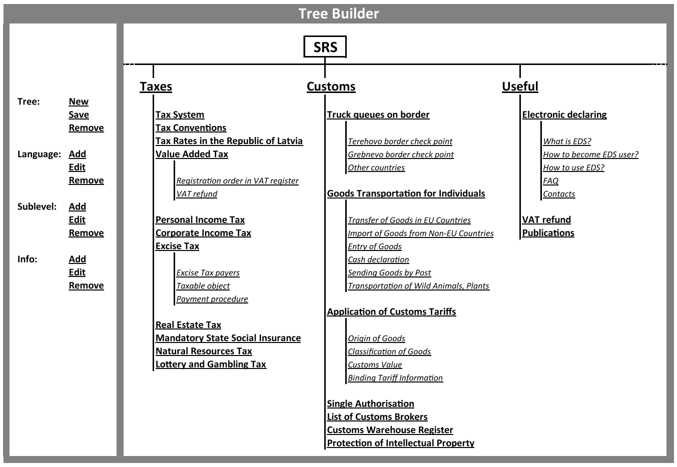


Fig. 4. Tree Builder of the SRS web site

Figure 4 represents a view of the Tree Builder component of WeSiMoTo during the development of Information Tree of the SRS web site.

5.2 Frame Modeling (Frame Composer)

Frame modeling gives the opportunity to combine Objects in different ways in order to produce Frames with concrete purposes or with particular shapes. As an example, Figure 5 represents the Advanced Search Frame of the SRS web site. It consists of Text, Picture, Free Input, Drop Down and Action Objects.

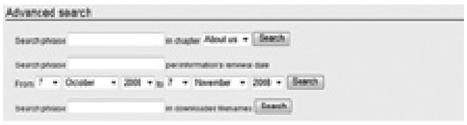


Fig. 5. Advanced search Frame of the SRS web site

Frame Composer is the component in WeSiMoTo that supports Frame modeling. It contains all predefined Objects and offers the following functions:

- create new / edit / remove Frame;
- set size parameters for the Frame;
- pop-up / overlaying Frame;
- add / edit / remove Object;
- save Frame in repository.

Figure 6 represents the Advanced Search Frame of the SRS web site previously shown in Figure 5, only here it gives a view of the Frame through the Frame Composer component of WeSiMoTo.

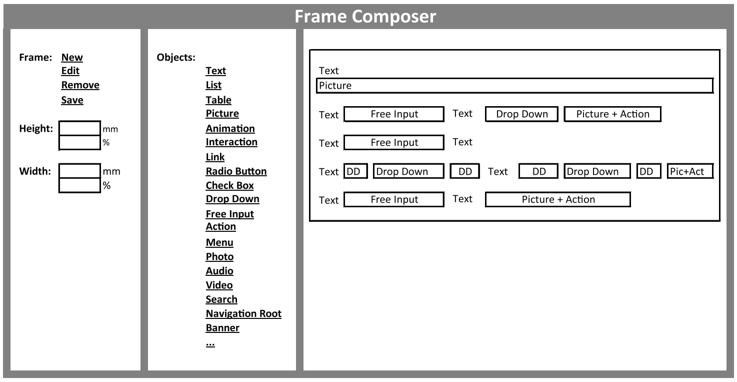


Fig. 6. Advanced search Frame of the SRS web site in Frame Composer

5.3 FrameSet Modeling (FrameSet Designer)

The function of FrameSet modeling is to prepare the general layout of a web site; it defines the number of columns and rows in a view, their height and width. FrameSet presents a logical and physical structure indicating where Frames then can be allocated in the browser window.

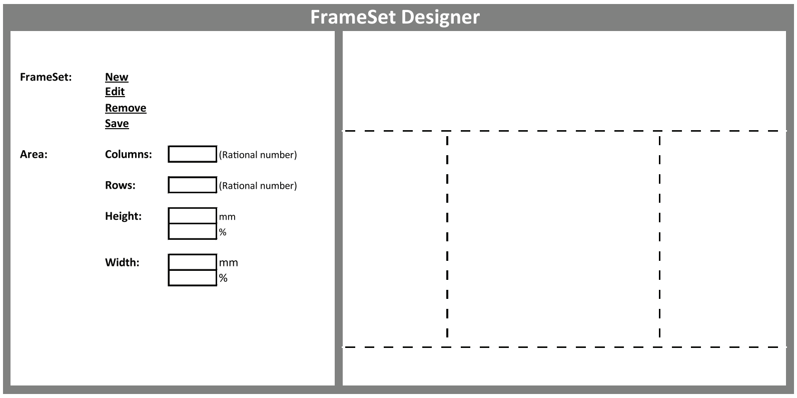


Fig. 7. FrameSet Designer

FrameSet Designer represented in Figure 7 is the component in WeSiMoTo that allows to make as many different FrameSet layouts as needed. Designer offers the following functions:

- create new / edit / remove FrameSet;
- set number of columns / rows;
- set size parameters for each area – fixed or proportional;
- save FrameSet in repository.

5.4 Navigation Modeling (Navigation Net)

Navigation modeling is the essence of web site modeling. It bounds together all parts of web site modeling described before. It allows to construct the net of relations among Information Tree, FrameSets, and Frames, permitting web site navigation.

Navigation Net is the component in WeSiMoTo that takes Information Tree as the basis and allows to choose any prepared FrameSet layout and connect it to a chosen branch or leaf of the tree. Then each FrameSet can be filled with Frames, meaning predefined Frames can be placed into FrameSet areas and connections among them are made. Consequently, navigation is built up.

General functions of Navigation Net are:

- show and navigate Information Tree;
- show and make available all predefined Frames;
- for each tree branch or leaf – attach / show / remove FrameSet;
- for each FrameSet area – add / remove Frame;
- add / edit / remove links from / to Frames and Information Tree;
- save model.

Figure 8 to Figure 11 demonstrate an example of navigation through the SRS web site. Figure 8 is the first in the row where Navigation Net describes the position corresponding to the main page of the SRS web site.

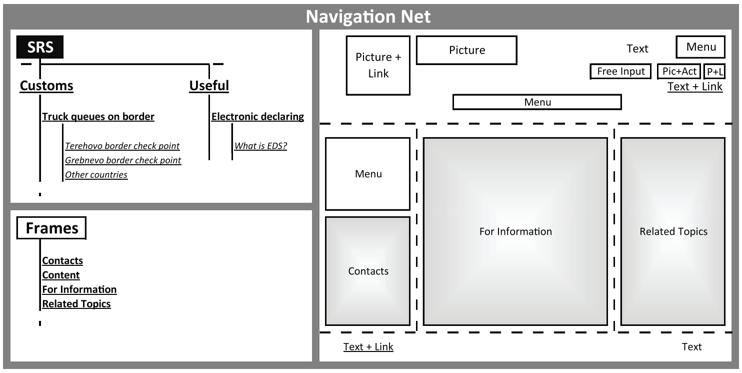


Fig. 8. Navigation Net representing the main page of the SRS web site in English

A Navigation Net window consists of three major parts: Information Tree, Frames, and FrameSet views. Since the space is limited, part of Information Tree can be viewed and the actual page is marked black. Frames are organized in a tree form and are accessible from Frames view. The right side is left for FrameSet construction, filling them with Frames and building up navigation among them all.

We are your partners in voluntary compliance!
For information
Information telephone numbers

Fig. 9. The main page of the SRS web site in English

Figure 9 represents the same English language main page of the SRS web site, but now in shape of a real site, where all Objects and FrameSet are replaced with actual textual and graphical information.

In order to see how the navigation works, a click on “Related Topics” – “Electronic Declaring” is simulated. Figure 10 represents the actual state after the click. Again, the root to the actual page is marked black, namely “SRS” -> “Useful” -> “Electronic declaring” -> “What is EDS?” Consequently, there are changes in the FrameSet side, where the previous is substituted with the corresponding to the new state in the Tree. FrameSet now has a different set of Frames, their location, and one column less.

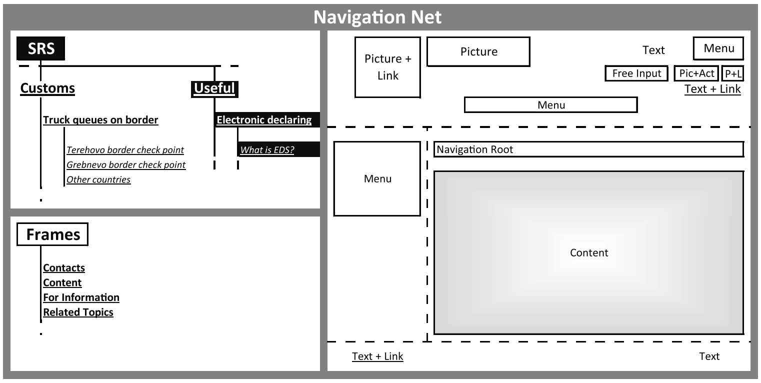
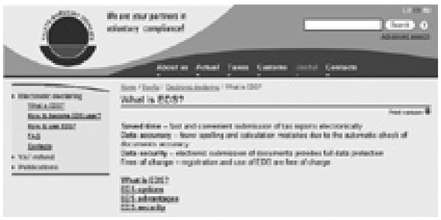


Fig. 10. Navigation Net representing EDS page of the SRS web site in English

Figure 11 represents the same English language EDS page of SRS web site, again in a shape of the real web site, where all Objects and FrameSet are replaced with actual textual and graphical information. This example shows the way the Navigation Net can be used for building up the navigation of the web site and how a prototype is build from the model and later simulated.


We are your partners in voluntary compliance!

About us | Actual | Taxes | Customs | Useful | Contacts

[illegible] / What is EDS?

What is EDS?

Saved time – fast and convenient submission of tax reports electronically
Data accuracy – fewer spelling and calculation mistakes due to the automatic check of documents accuracy
Data security – electronic submission of documents provides full data protection
Free of charge – registration and use of EDS are free of charge

What is EDS?
EDS options
EDS advantages
EDS security

Fig. 11. EDS page of the SRS web site in English

5.5 Web Site Model

Web Site Model is a set of internally coherent four types of modeling processes mentioned before. All types of processes, except Navigation Net, and their names are presented in a tree-type structure that shows the content of the model.

The model's content consists of three parts:

- Information Tree's diagram – presents all information in a structured way, and it is unique to the whole model;
- Frames' repository – every row describes one Frame with attached design that shows the content of the Frame – Objects in it and its parameters;
- FrameSets' repository – each row describes one FrameSet with attached layout of the FrameSet and its parameters.

6 Conclusions and Further Directions

This paper examined the domain-specific web site modeling language WeSiMoLa and the web site modeling tool WeSiMoTo based on the WeSiMoLa language, which give the possibility to collect, develop and store requirements in an easily understandable way for both customers and suppliers. In contrast to the waterfall principle, the gathering of requirements here is organized on the basis of gradual elaboration approach. Moreover, a web site model can be built from the requirements. A prototype of the web site is the consequent next step that can be executed, thus clearly demonstrating the planned web site in action and assuring both parties of how common their expectations are and whether they will be met. Prototyping is considered as a natural extension of this research paper.

The proposed method suits well the need for acquiring requirement specifications for various kinds of simple type web sites such as HTML-based home pages, promotional web sites, and others of the same sort. The approach described in the paper also solves well the requirement management for web portals based on content management systems. The scope of this research does not include web sites integrated with sophisticated data sources, because those would involve an additional definition of DSL objects for the data storage. That gives an opportunity for further and broader research within the field of web development.

The proposed web site modeling language is built upon practical experience and requirements, therefore it can't be perceived as fixed and complete. Primarily it applies to the types of Objects that are used in the Frame composition process. Evolution of the web will bring ever more sophisticated object types that would need to be included into the specification language.

However, the example of the already existing SRS web site and its model in WeSiMoTo represents how user friendly is the modeling language and how natural is the modeling tool in operation. The advantage of the described language is that in contrast to a general-purpose modeling language such as the UML, WeSiMoLa (with supporting WeSiMoTo) allows a particular type of issues or their solutions on the Web to be expressed more clearly.

In the future this modeling language and the related tool will be investigated more by applying them in different new web site development projects at the requirement analysis stage, thereby achieving greater approbation and higher precision in each of their components. Secondly, this paper does not describe the collaboration between the web site and Content Management System (CMS) and its formalization, which is for a prospect of further research in the future.

References

1. Lowe D., Eklund J. (2002) Client Needs and the Design Process in Web Projects, *Journal on Web Engineering* 1, Rinton Press, pp. 23–36.
2. Sommerville I., Ransom J. (2005) An empirical study of industrial requirements engineering process assessment and improvement, *ACM TOSEM* 14, pp. 85–117.
3. Unified Modeling Language (UML), version 2.1.2., available online: *http://www.omg.org/technology/documents/formal/uml.htm*.
4. Business Process Modeling Language (BPML), available online: *http://www.service-architecture.com/web-services/articles/business_process_modeling_language_bpml.html*.
5. State Revenue Service of Republic of Latvia, available online: *http://www.vid.gov.lv*.
6. Van Deursen A., Klint P., Visser J. (2000) Domain-Specific Languages: An Annotated Bibliography. Computer Based Learning Unit, University of Leeds, available online: *http://homepages.cwi.nl/~arie/papers/dslbib/*, February 2000.
7. Herman I. (2008) W3C Semantic Web Activity. Available on the internet *http://www.w3.org/2001/sw/*. April 2008.
8. Jacobs I. (2008) About the World Wide Web Consortium (W3C). Available online: *http://www.w3.org/* Consortium/, February 2008.
9. Swartz A. Application/rdf+xml Media Type Registration", available online: *http://www.ietf.org/rfc/rfc3870.tx*t, September 2004.
10. Schreiber G., Dean M., van Harmelen F., Hendler J., Horrocks I., McGuinness D. L., Patel-Schneider P. F., Stein L. A. OWL Web Ontology Language Reference/W3C Recommendation. Available online: *http://www.w3.org/TR/owl-ref/*, February 2004.
11. Universal Networking Digital Language Foundation (UNL). Available online: http://www.undl.org/.
12. Manukyan Av., Manukyan Ar., Mailyan A., Sayadyan A. (2008) Website Parse Templates. Available online: *http://tools.ietf.org/html/draft-manukyan-website-parse-templates-00*, April 2008.
13. The mission of the ESSI WSMO working group. Available online: *http://www.wsmo.org*.
14. De Bruijn J., Bussler C., Domingue J., Fensel D., Hepp M., Keller U., Kifer M., Kopecky J., Lausen H., Oren E., Polleres A., Roman D., Scicluna J., Stollberg M. (2005) Web Service Modeling Ontology. Available online: *http://www.w3.org/Submission/WSMO/*, June 2005.
15. Web Service Modelling eXecution environment. Available online: *http://www.wsmx.org/*.
16. De Bruijn J., Fensel D., Keller U., Kifer M., Lausen H., Krummenacher R., Polleres A., Predoiu L. (2005) Web Service Modeling Language. Aavailable online: *http://www.w3.org/Submission/WSML/*, June 2005.
17. Ceri S., Fraternali P., Bongio A. (2000) Web Modeling Language: A Modeling Language for Designing Web Sites. In: Proc. of the 9th Intl World Wide Web Conference, May 2000, Amsterdam, pp. 137–157. Available online: *http://www9.org/w9cdrom/177/177.html*.
18. Koch N., Kraus A. (2002) The Expressive Power of UML-Based Web Engineering. In: Proc. of the Second Intl Workshop on Web-Oriented Software Technology (IWWOST02), Malaga, 2002, pp. 105–119. Available online: *http://www.pst.ifi.lmu.de/projekte/uwe/*.
19. Arnicans G., Karnitis G. (2006) Intelligent Integration of Information from Semi-Structured WEB Data Sources on the Basis of Ontology and Meta Models. In: Proc. of the 7th Intl Baltic Conference DB&IS, Vilnius, 2006, pp. 177–186.