EG UK Theory and Practice of Computer Graphics (2006)
M. McDerby, L. Lever (Editors)

GraSSML: Smart Schematic Diagrams, XML and Accessibility

Z. Ben Fredj and D.A. Duce

Department of Computing, Oxford Brookes University, UK

Abstract

This paper describes work in progress at Oxford Brookes University. The project called Graphical Structure Semantic Markup Languages (GraSSML) aims at defining higher-level diagram description languages for the World Wide Web, which capture the structure and the semantics of a diagram and enable the generation of accessible and "smart" presentations in different modalities such as speech, text, and graphics. GraSSML is broken down into three levels: semantics, structure and presentation. Each of these levels captures a specific aspect of a diagram. The semantic level language is highly dependent on the type of diagram considered and the knowledge of the domain (ontology) in which it is used. Using the proposed approach, the structure and the semantics of the diagram is made available at the creation stage. The availability of this information offers new possibilities allowing Web Graphics to become "smart". The paper outlines the relevant limitations of SVG and some approaches aiming to resolve the problem of graphic accessibility. It then describes our approach in addressing some of these limitations and presents the new possibilities that these smart graphics lead to.

Categories and Subject Descriptors (according to ACM CCS): H.5.3 [Group Organization Interfaces]: Web-based interaction I.3.6 [Methodology and Techniques]: Languages K.4.2 [Social Issues]: Assistive technologies for persons with disabilities

1. Introduction

Graphics are a powerful way of conveying information. Until now, the use of graphics has made life much easier for most sighted users, but people with disabilities or users who work in environments where visual representations are inappropriate cannot access information contained in graphics, unless alternative descriptions are included.

Diagrams are a fundamental component in the exposition of scientific research and are unavoidable in professional life. Despite their importance, so far not much has been done to provide accessibility support to information of this kind. But things are about to change. Access to graphical information has now become a very important issue, and even more so, since the report on Web accessibility from the UK's disability Rights Commission in 2004 [Dis04]. Part III of the 1995 Disability Discrimination Act places legal obligations on organizations to provide equal access to information [BBC04].

It is important to understand that accessibility is not only for people with obvious disabilities but also for people who simply access information and learn in different ways. To allow full access to the web it is also important that people with disabilities can create accessible web content containing accessible web graphics.

For a long period of time web graphics were supported through raster based bitmap formats such as GIF, PNG and JPEG. These formats present several limitations which are overcome by vector formats [BFD03]. A review of Web file formats appears in [DHH02].

The emergence of SVG [W3C01a] has changed the way 2D graphics are created on the web. SVG offers many advantages and has introduced accessibility features that raster formats do not offer [MK00]. It describes pictures in terms of geometrical primitives from which they are composed (i.e. paths and text). So, when using SVG, information about the diagram is available to the browser in terms of the objects it is composed of.

SVG is a significant step forward in improving graphics

on the Web but presents a number of limitations as it only captures diagrams at a low level of abstraction. It is more a "final form" presentation, which involves some drawbacks in the direct creation, modification and access of complex, highly structured, diagrams. In spite of this new standard, the web is still suffering from a lack of techniques to make graphical content accessible.

The next section describes the problem in more detail and presents the limitations of SVG and of some of the approaches previously taken to resolve the problem of graphic accessibility. Section 3 outlines the authors' approach to the problem. The following section discusses the current state of the GraSSML development and system architecture. The final section contains conclusions and thoughts on future work.

2. Problem and Related Work

2.1. Advantages and limitations of SVG

SVG presents many advantages and provides many accessibility benefits [MK00]: some originating from the vector graphics model (scalable, zoomable and plain text format which is searchable), some inherited because SVG is built on top of XML (interoperability, Unicode support, CSS, XSL, DOM, wide tool support, etc.) and some in the design of SVG itself (alternative equivalent). However, some important issues still remain to be addressed.

SVG does not capture diagrams at a high level of abstraction. It is a "final form" presentation, which involves some drawbacks in the direct creation of complex, highly structured, diagrams. It is difficult at this level to handle the resizing and positioning of different shapes in complex diagrams. A simple modification such as changing the alignment from vertical to horizontal can be awkward. SVG does not allow flexible readjustment of layout in response to viewer requirements and the viewing environment, such as different screen formats.

The difficulty is that the intentions of the author are not totally captured. The structure of the resulting SVG may reflect the sequence of operations used to create the diagram, rather than the intrinsic object structure within the diagram itself. This is likely to be the case for diagrams created with a general-purpose drawing tool.

Although SVG stores structural information about graphical shapes as an integral part of the image and allows metadata to be attached to primitives, there is little real scope for generating alternative presentations from the description at this level. An additional issue is that an SVG document contains the semantics of the diagram only implicitly. The "alternative equivalents", which allow the author to include a text description for each logical component and a text title to explain the component's role in the diagram, could become tedious for the creation of complex diagrams. If the metadata added by the author are not accurate enough, the semantics of the diagram could differ from the description obtained from the metadata. The following section presents the main approaches aiming at making graphics accessible, some exploring the accessibility features of SVG.

2.2. Bottom-up Approaches to make Graphics accessible and their limitations

Many works have explored different methods to make graphics accessible to blind or visually impaired people by representing graphics through an auditory interface, tactile drawings, text description, etc. Although the proposed approaches partially address the accessibility problem of graphics, they present many limitations. Some of these approaches are presented below and their limitations are discussed.

The two following projects (BIS) and (GUIB) require active human intervention.

The "Blind Information System" (BIS) is an interesting project, developed at the Czech Technical University [MS99]. It is a system for the interpretation of graphical information. It develops a hierarchical picture description methodology based on an XML grammar which characterizes pictures by structure and semantics. The result obtained is a text (XML) document describing the semantic and structural view of the picture. The description obtained depends on the person creating it. This person, who will probably not be the author of the diagram, might omit important information from the description.

In the context of the "Graphical User Interfaces for Blind People" (GUIB) project [K*95] a two phase methodology is introduced, which is supported by software which allows the work of one sighted person to be used by many blind people. The two phases are:

- Phase 1: a modeling tool supports a sighted person (the moderator) in producing a model of the visual graphic taking into account the representational and presentational aspects of the graphics and the interaction facilities available to the blind person.
- Phase 2: a presentation tool allows the blind person to interact, explore and experience the graphics without the assistance of a sighted person. This approach presents an obvious drawback: the moderator, who most of the time is not the author of the graphics, has an important responsibility. He decides what information to convey and imposes his view when the graphic is being read.

The TeDUB (TEchnical Drawings Understanding for the Blind) project which began in 2001, explored the possibility of a semi-automatic analysis of diagrams [H*04]. The system developed generates descriptions of certain classes of graphics automatically (electronic circuit diagrams, UML diagrams and architectural plans) and allows blind people to

read and explore them on a PC with the aid of a computer games joystick and a screen reader. The graphics used in the TeDUB project are taken from different sources. They can be bitmap graphics, vector graphics (SVG), file format with semantic content (e.g. XMI, CAD format). The system is composed of two main parts. The first is the "DiagramInterpreter" which is the knowledge processing unit. It analyses the diagram provided, operates on a network of hypotheses until a semantic description of the whole diagram is found and converts it into a representation that is used by the second main part of the system: the "DiagramNavigator". The latter allows the blind users to navigate and annotate these diagrams through a number of input and output devices. In the event that the automatic process might lead to wrong results, the "ImageAnnotator" is used to make corrections.

This approach for the presentation and navigation of the graphical information offers many advantages for blind users. But the (semi-) automatic analysis of the diagram information might produce wrong results and might need active human intervention and time.

Some research groups have explored the accessibility features of SVG and have attempted to address some of the SVG limitations. Their approaches present some solutions but also demonstrate that not all SVG limitations can be overcome at this level.

The Science Access Project (SAP) group at Oregon University aims at developing methods to make information accessible for people with print disabilities. In 1997, SAP started a research project to make bitmap graphics accessible. They obtained some useful results and acquired an understanding of the possibilities and limitations of audio/tactile access of bitmap graphics. With the emergence of SVG they stopped this research project concluding that "there is no way to make bitmap graphics that are more-or-less automatically accessible, a separate accessibility page needs to accompany each graphic" [G*02]. They then developed the ViewPlus project [BG04]. The project current hardware/software product line has a group of SVG applications which can provide excellent access "to most" SVG graphical information for all. The SAP research group has successfully explored many accessibility features of SVG but it also has identified some of its limitations [GB01]. Some SVG documents become less accessible when created without <title> and <desc> elements, so to overcome these drawbacks SAP has created an SVG editor that permits addition of a title and description to elements that do not have them. Some SVG documents are very badly structured and therefore less informative. So occasionally, re-ordering the hierarchical structure of the SVG to organize objects into proper groups becomes necessary. The ViewPlus project has a good approach by exploring the information behind the picture but the solutions proposed to overcome SVG limitations need too much effort in adding information and reorganizing it. It can be tedious and time consuming. It illustrates the fact that SVG is too low level and not informative enough regarding the structure of the graphical information.

An extension to SVG, called Constraint Scalable Vector Graphics (CSVG) [BTM*01], has been proposed. It partially addresses some of the SVG limitations by proposing semantic zooming, differential scaling and semantics preserving manipulation. These additional capabilities allow alternate layouts for the same logical group of components in a diagram, which greatly improves SVG's value. But whilst permitting a more flexible description of figures, CSVG remains very close to SVG and still captures diagrams at a similar low level of abstraction.

The SVG linearizer tool developed initially by Lovet and Dardailler [LD00] and then continued by Herman and Dardailler [HD02] is one approach to the accessibility problem. The idea is to generate a textual linear representation of the content of an SVG file by using a metadata vocabulary describing it. The author has to describe the SVG content using this RDF vocabulary and to add textual descriptions to all elements that constitute primary RDF resources. Then, from this information plus information contained in the SVG file itself, an HTML file is generated. This operation can be tedious and not very efficient in very complex diagrams and adding the RDF annotations is an onerous task for the author. This method is too dependent on the creator's patience and willingness to produce appropriate metadata.

David Dodds presents a different technique to explore the content of SVG [Dod03]. He presents a way of providing a linguistic capability in an SVG system using a program which is capable of performing the necessary part of an ascription system. The SVG file is actively scanned by the program (SVG picture itself, metadata about the picture, other non-explicit knowledge) and analyzed. The user can then query the system to obtain information about the picture expressed in natural language. SVG being low level, this approach involves a certain amount of computation.

2.3. Problem

These previously presented approaches that we have categorized as "bottom-up approaches" have been looking at the problem upside down. They start with the graphical representation of the diagram. The diagram is analyzed and interpreted by a predefined system and/or a moderator (who is not usually the creator of the diagram). The latter is required to add "metadata" to help understand the information. Some of these approaches involve writing difficult programs in order to "discover" the structure or semantics of the graphic. Many limitations involved in these approaches could probably be overcome if the information on the structure and the semantics of the graphics were made part of the graphics.

In 1997, John A. Gardener [G*97] gave an overview of the concepts of "smart graphics" (information behind a picture) and intelligent graphics browsers for accessing such in-

formation. He highlighted the fact that *"Nearly every part of smart graphics technology exists today, but to our knowledge there is no complete package that incorporates everything necessary to author a smart picture, incorporate it into an electronic document, and display it intelligently"*.

The GraSSML system described in this paper aims, by means of its family of languages, to be such a package by providing access to this most valuable "information behind the diagram". The following section presents this approach in more detail.

3. The GraSSML Approach

3.1. The idea: "Little Languages"

The idea is to reduce a task to a sequence of transformations between inputs and outputs expressed in different "Little Languages" in Bentley's phrase [Ben86]. Bentley argues that a language is "any mechanism to express intent, and the input to many programs can be viewed as statements in a language". By viewing inputs in this way, it is possible to see how to decompose a computation into a sequence of processing steps in which the output of one step is fed into the input of the next. The main example in his column is Kernighan's PIC language [Ker82], a component of the Unix document production suite for constructing line drawings from textual descriptions. This language works by generating code that is interpreted by the TROFF typesetting system to produce the drawing on the page. He shows how other Little Languages designed to express particular kinds of graphical representations, for example chemical structures and graphs, can be translated into PIC and hence rendered in a document (e.g. GRAP and CHEM).

The aim of GrassML is not to define an XML language that is an exact analogue of PIC but rather to draw inspiration from PIC in defining a language at the structure level based on the key ideas in PIC.

3.2. The GraSSML family of Languages

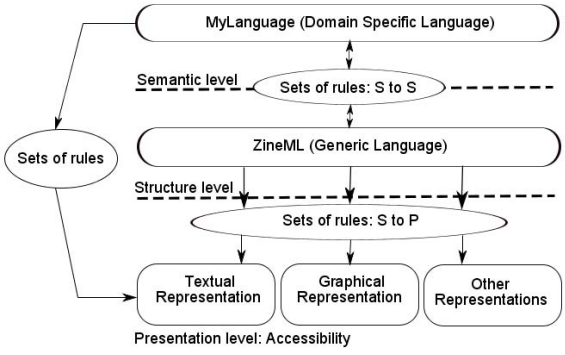

Figure 1: *The Levels of GraSSML.*

GraSSML is organized into three levels (Figure 1):

- Semantic level: the semantics of the diagram is expressed using the language called MyLanguage.
- Structure level: the structure of the diagram is expressed using the language called ZineML which can also be based on the syntax of the diagram.
- Presentation Level: the representation of the diagram in different modalities, such as graphical, textual, auditory or any other representation. SVG will be used as the graphical output renderer at this level.

3.3. Structure and semantics of diagrams

An important aspect of this project is concerned with the syntax and semantics of diagrams.

In natural language, words can be mapped into a set of meanings whereas in a visual language, geometric objects do not have a unique semantic interpretation. There are a very large variety of diagrammatic notations [Tuf83, Tuf90]. There are no universal visual conventions and each person interprets graphical information using his own mental schema and/ or imagination. However, for a group of users having the same background and educational level it should be possible to identify some similarities concerning their mental schema.

Much research has been done to explore the analogies between diagrams and textual representations. This can be divided into three parts: syntax, semantics and pragmatics. The Human Communication Research Centre at the University of Edinburgh [Gur99] is exploring a theory of diagrammatic communication by studying the similarities and differences between diagrammatic systems and textual systems of communication.

So "just as we have to learn the language we read and write, we have to learn the representational conventions of diagrams. Once learned, the process of reading or interpreting a diagram is relatively effortless" [FKA*92]. But unfortunately for a computer life is not so easy. Indeed, a computer needs a set of formal representational conventions to carry out this analysis. It is in this context that The Biological Knowledge Laboratory at Northeastern University is using Graphics Constraints Grammars for describing and analysing diagrams in technical documents [FKA*92].

Mark Minas from the Erlangen-Nürnberg University (Germany), defined a diagram language as a formal language that can be defined by its syntax, semantics and pragmatic [Min99]. He introduces a method that translates a diagram into an abstract semantic description. The method is based on a specification of the translation process. The diagram, which is represented as a set of atomic components, is translated into its semantic representation as a hypergraph. Hypergraphs appear to be an appropriate way to represent diagrams on a different level of abstraction. The translation process consists of two steps:

- The specification of spatial relationships between the components of the diagram.
- A translation grammar, which describes the diagram syntax and the rule for generating the semantic.

Each field has its own notation, syntax and semantic rules. The approach taken by Minas is a starting point for the highest level of the GraSSML family of languages. Before generating a semantic representation of a diagram in a specific field, we need to know the rules of this field concerning the syntax and the semantics of each component. So the method proposed by Minas could be the solution when used in reverse for deriving rules by analyzing a wide range of diagrams in a specific field following the two steps described above.

A set of semantic rules needs to be defined. A diagrammatic semantic grammar should be developed to address this issue. Following this specific predefined grammar, the user should be able to define the semantics of diagrams. This diagrammatic semantic grammar will be based on new technologies emerging in the context of the Semantic Web activity.

3.4. Taxonomy of diagrams

There is an infinite range of possible shapes that people can imagine. The investigation of different varieties of diagrammatic notations is one of the interests of researchers in psychology and cognition. Various classes of graphics can be identified depending on the chosen classification scheme. Different diagrams types are looked at in different ways. The same type of diagram can even be read in a number of different ways. A wide range of classifications has already been described in several different academic fields. The structural taxonomy proposed by Lohse [LBWR94] has been recruited to identify some classes of diagrams (e.g. process diagrams, hierarchical diagrams, etc.) for which XML representations might be created. In the first stage of this work we are focusing on structured diagrams and developing ZineML to cover this class of diagrams.

3.5. Accessibility

The availability of the information "behind the graphic" allows us to generate alternative presentations. To create such presentations, we refer to studies on what kind of information is needed and how to describe graphics textually or verbally [Web, BE98, CK01].

4. GraSSML system

4.1. Structure level: ZineML

"ZineML" aims to be at a higher level of SVG by representing the structure of the diagram. It facilitates the creation and modification of diagrams. The code aspires to be readable by humans and to give a good overview of the diagram structure. The language seeks to be rich enough to allow accessible alternatives of the structural representations of the diagram.

The initial version of ZineML is being designed to cater for the needs of the class of structure diagrams such as process diagrams and organizational diagrams. It proposes a set of basic shapes selected to cover a wide range of possibilities (Figure 2) for structure diagrams common in different fields (e.g. business, computing, biology). Starting with a predefined library of diagrams which have been classified, the set of basic shapes used in each diagram is then extracted. It is important to keep in mind that it is impossible to find all possible shapes since the diagrammatical notation has no limit. Indeed, the user often imagines and creates his specific own shapes. So ZineML should be extendable, and allow the addition of new shapes defined by the user.

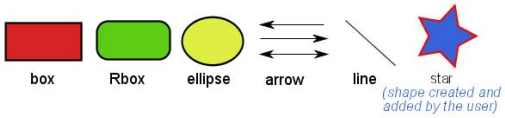

Figure 2: *Examples of basic shapes defined by ZineML.*

At this level of abstraction the language "ZineML" is not field dependent, but proposes some options to express and apply rule sets when creating diagrams. These rule sets could be used to tailor the diagram specifically to a domain.

ZineML offers the possibility to determine and to adjust positions and sizes semi-automatically, with a minimum of effort from the author for creation and modification of a diagram. The graphical representation of ZineML is done in accordance with the user predefined rule sets (Figure 1) called "Set of rules S to P" (Structure to Presentation). These depend on the method used to create the diagram. The following options have been taken into account for the design of ZineML.

- The author wants to create a diagram without giving any explicit size and positional information. In this case a default algorithm defines default size and positions for each element of the diagram. For example, each shape may adjust itself to its content, which can be a text string or any other shape. If the default diagram obtained does not suit the user, he still has the possibility of changing the rules applied by selecting another presentation algorithm or by using one of the following options that requires the user to give more explicit information on how he would like the structure to be presented.
- The author uses elements provided by ZineML to specify positions and sizes of the elements. For example, the attributes "bottom", "top", "left" etc. for the alignment, or "h" and "w" for the height and width of a shape, the elements <down>, <up>, <right>, <left> for the drawing direction.

- The author creates the diagram with a specific set of global constraints. The specification of these constraints allows an author to define the syntax that all valid diagrams of this domain have to conform to. These constraints are defined in the set of rules used to represent the diagram: size of graphical element, relative positions, valid relations etc.

Whatever the option used, if any modification is made, the presentation algorithm maintains the structure of the diagram by reapplying itself on the modified ZineML file, respecting the rule sets defined.

4.2. Semantic Level: My Language

"MyLanguage" is the language used at the level above ZineML to capture the semantic intent behind the diagram. It does not aim to be universal at this level. It has to be applied to a specific area where clear conventions are followed when creating diagrams (e.g. Organization charts, UML, Merise, QOC method, State transition diagrams).

MyLanguage does not make any commitment to graphical presentation, but aims to capture the concepts and relationships between concepts that are to be expressed in pictorial or other representational form.

For example, in the case of an Organization hierarchy language "Org_Hierarchy" (MyLanguage) might express the notion that "an employee reports to his manager and a manager manages an employee".

MyLanguage is based on predefined syntax and semantic rules expressed by ontologies. Each ontology attempts to formulate concepts and their relationships in a specific field (e.g. business organizations, networking, business exchange, etc.).

Although MyLanguage does not make a commitment to a particular kind of presentation, in order to develop languages that will be translated into particular kinds of diagrams, it is necessary to first study the field and the class of the diagrams with the aim of discovering what concepts and properties these diagrams seek to represent in the field.

It should be understood that MyLanguage is field-dependent, hence the name!

At this level "Who knows better than the authors"? The information concerning the different notations used, the possible relationships between them and the meaning behind each of them, is the information which is required in order to create such a language.

It should be borne in mind that "A diagram is worth a thousand words". Indeed, the semantics of the diagram can be complex, and without the help of the author the interpretation generated may be both verbose and shallow in the sense that the author is the one who knows "exactly" what the main message behind the diagram is and therefore can give concise information.

The XML language obtained is an intuitive domain specific language; it employs the notations and concepts familiar to practitioners of the domain and by doing so it makes the semantics explicit to the user. Consequently a domain expert can easily understand the information. Even if all the information needed has been gathered, an important issue remains unsolved: the semantic is only implicit for the machine. Indeed, XML covers only the syntactic level and does not provide any means of talking about the semantics of data. There is a need to make these semantics explicit to the computer.

There is an obvious link to activity in the Semantic Web area concerned with the expression of subject ontologies and relationships expressed over terms defined in ontologies.

RDF forms the core of the Semantic Web. It is a framework for describing "resources" and relations between them. RDF is domain-independent. RDF Schema provides a framework to describe application-specific classes and properties. RDFS and OWL allow the specification of ontologies. They both provide a vocabulary for describing properties and classes of RDF resources [W3C01b].

Current work is exploring the creation and use of ontologies to make the semantics of the diagrams explicit. Having the semantics of the diagram available makes it possible to express queries concerning specific parts of a diagram in novel ways: "smart diagrams" become a reality.

Along with the specification of a particular MyLanguage, notational conventions are created. The notational conventions govern the diagrammatic presentation of the elements of MyLanguage (see Figure 1: Set of rules "S to S"), and are captured in sets of rules governing the generation of ZineML from MyLanguage. For example, in the case of "Org_Hierarchy" (MyLanguage):

- <Employee> (which is an element in "Org_Hierarchy") will be represented by a box with a centered label;
- <ReportsTo> will be represented by a dotted arrow connecting the source and destination entities.

4.3. Presentation level: e.g. SVG and XHTML

At this level, the information on the structure and semantics of the diagram allows alternative representations to be generated:

- Graphical Representation: SVG is used as the graphical output renderer. For complex diagram we are aiming at allowing an interactive exploration of the diagram by using some of the SVG facilities and the available information concerning the structure and semantics of the diagram (the user can hide some details).
- Textual Representation: a verbalization model has been

implemented. It allows the generation of a textual representation of the structure and the semantics of the diagram.
- Query System: at a later stage a query system will be developed based on the explicit structure and semantics of the diagram.

5. Implementation

To prove the feasibility of the approach a prototype is being developed. The architecture of the prototype system is shown in Figure 3. Implementation of ZineML for two classes of diagrams: process and organizational diagrams is in hand. The Java Programming Language has been used to implement the presentation algorithms aiming at generating the graphical representation of the diagram (SVG). The grahical representation of the MyLanguage document is obtained by applying an XSLT transformation which maps MyLanguage content to the corresponding ZineML representation according to predefined rule sets. The SVG representation of the resulting ZineML document is generated by a second XSLT transformation which embeds another set of predefined rule sets.

The verbalization model allowing the generation of the structure and semantic of the diagram based on the syntax of the corresponding XML language used (ZineML or MyLanguage) has been implemented using an XSLT Transformation on the ZineML and the MyLanguage languages. The output is an XHTML file containing the textual representation of the structure and the semantics of the diagram.

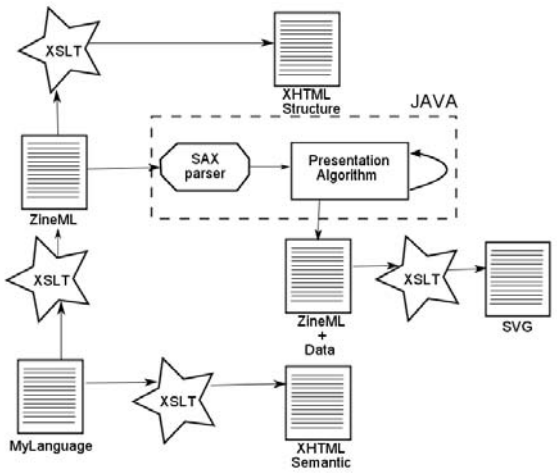

Figure 3: *The GraSSML architecture.*

The other functionalities of GraSSML are at the design stage ready for implementation. Once the implementation is completed, a set of experiments will be carried out in order to evaluate the usability and accessibility of the system. The results will then be compared with other existing techniques.

6. Conclusion and future work

Graphics on the Web have improved significantly but a number of issues still remain unsolved. This paper has outlined these issues and has presented some related work aiming at resolving them. These only address some of the problems and they emphasize the need to make the graphical information (information on the structure and the semantics of the graphics) part of the graphic.

The authors' approach has looked at the problem of creation, modification, access and exploration of web graphics from a different perspective. Using the GraSSML system, the graphical information is made available at the creation stage by the author. The availability of this information offers new possibilities. Web Graphics become "smart", they carry their knowledge with them, intrinsically they know their structure (ZineML) and semantics (MyLanguage), and they could also know who created them and how, when and why they have been created. Any application smart enough to access explore and present this knowledge in the right way should be able to propose new solutions for identified problems (adaptive and accessible graphics). This could substantially improve the accessibility of web graphics. GraSSML could be the starting point for many projects currently aiming to access, present, explore and adapt graphical information [MMS04, Duk04].

Acknowledgements

Financial support for ZBF from Oxford Brookes University is gratefully acknowledged.

References

[BBC04] BBC NEWS REPORT: A web open to all? http://news.bbc.co.uk/1/hi/technology/3629599.stm, 2004.

[BE98] BENNETT D., EDWARDS A.: Exploration of Non-seen Diagrams. In *ICAD'98* (1998).

[Ben86] BENTLEY J.: Little Languages. *CACM 29*, 8 (1986), 711–721.

[BFD03] BEN FREDJ Z., DUCE D.: Schematic Diagrams, XML and Accessibility. In *Proceedings of Theory and Practice in Computer Graphics* (2003), IEEE Computer Society Press, pp. 49–57.

[BG04] BULATOV V., GARDNER J. A.: Making Graphics Accessible. In *SVG Open* (2004).

[BTM*01] BADROS G., TIRTOWIDJOJO J., MARRIOTT K., MEYER B., PORTNOY W., BORNING A.: A Constraint Extension to Scalable Vector Graphics. In *Tenth International World Wide Web Conference, WWW10, Hong Kong* (2001), pp. 489–498.

[CK01] CORNELIS M., KRIKHAAR K.: *Guidelines for Describing Study Literatrue*. FNB Amsterdam, 2001.

[DHH02] DUCE D., HERMAN I., HOPGOOD F.: Web 2D Graphics File Formats. *Computer Graphics Forum 21*, 1 (2002), 43–64.

[Dis04] DISABILITY RIGHTS COMMISSION: Formal investigation report: Web Accessibility. http://www.drc-gb.org/publicationsandreports/report.asp, 2004.

[Dod03] DODDS D.: Accessing SVG Content Linguistically and Conceptually. In *SVG Open* (2003).

[Duk04] DUKE D.: Drawing Attention to Meaning. In *Adaptive Displays Conference* (2004).

[FKA*92] FUTRELL R., KAKADIARIS I., ALEXANDER J., FUTRELL J., CARRIERO C., NIKOLAKIS N.: Understanding Diagrams in Technical Documents. *IEEE Computer* (July 1992), 75–78.

[G*97] GARDNER J. A., ET AL.: The Problem of Accessing Non-Textual Information On The Web. In *Proceedings of the 1997 Conference of the W3 Consortium, Santa Clara, CA* (April 1997).

[G*02] GARDNER J. A., ET AL.: Accessing Maps, Diagrams, and similar Object-Oriented Graphics. http://dots.physics.orst.edu/graphics.html, 2002.

[GB01] GARDNER J. A., BULATOV V.: Smart Figures, SVG, and Accessibility. In *Proceeding 2001 CSUN International Conference on technology and persons with Diabilites, Los Angeles, CA* (March 2001).

[Gur99] GURR C.: Effective Diagrammatic Communication: Syntactic, Semantic and Pragmatic Issues. *Journal of Visual Languages and Computing 10*, 4 (1999), 317–342.

[H*04] HORSTMANN M., ET AL.: TEDUB: Automatic interpretation and presentation of technical diagrams for blind people. In *CVHI* (2004).

[HD02] HERMAN I., DARDAILLER D.: SVG Linearization and Accessibility. *Computer Graphics Forum 21*, 4 (2002), 777–786.

[K*95] KURZE M., ET AL.: New Approaches for Accessing Different Classes of Graphics by Blind People. In *Proc. Of the 2nd TIDE Congress* (1995), Paris, Amsterdam: IOS Press, pp. 268–272.

[Ker82] KERNIGHAN B.: PIC- A language for Typesetting Graphics. *Software Practice and Experience 12* (1982), 1–21.

[LBWR94] LOHSE G., BIOLSI K., WALKER N., RUETER H.: A Classification of Visual Representations. *CACM 37*, 12 (1994), 36–49.

[LD00] LOVET G., DARDAILLER D.: SVG Linearizer tools. http://www/w3.org/WAI/ER/ASVG/, September 2000.

[Min99] MINAS M.: Creating semantic representations of diagrams. In *AGTIVE* (1999), pp. 209–224.

[MK00] MCCATHIENEVILE C., KOIVUNEN M.-R.: Accessibility Features of SVG. http://www.w3.org/TR/SVG-access/, 2000.

[MMS04] MARRIOTT K., MEYER B., STUCKEY P. J.: Towards flexible graphical communication using adaptive diagrams. In *ASIAN* (2004), pp. 380–394.

[MS99] MIKOVEC Z., SLAVIK P.: System for Picture Interpretation for Blind. http://cs.felk.cvut.cz/~xmikovec/bis/interact99/, 1999.

[Tuf83] TUFTE E.: *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[Tuf90] TUFTE E.: *Envisioning Information*. Graphics Press, 1990.

[W3C01a] W3C: Scalable Vector Graphics (SVG) 1.0 Specification. http://www.w3.org/TR/SVG/, 2001.

[W3C01b] W3C: Semantic Web. http://www.w3.org/2001/sw/, 2001.

[Web] WEB ACCESSIBILITY FOR ALL: How to create Descriptive Text for Graphs, Charts and other Diagrams. http:www.cew.wisc.edu/accessibility/tutorials/descriptionTutorial.htm.