Authoring Semantic Annotations for Non-Visual Access to Graphics

Tomas Murillo-Morales, Klaus Miesenberger
Johannes Kepler University, Linz (Austria)

Jaromír Plhák
Masaryk University, Brno (Czech Republic)

Tomas.Murillo_Morales@jku.at, 60773@mail.muni.cz, Klaus.Miesenberger@jku.at

Abstract

Semantically-enhanced graphics are annotated with formal underpinnings in order to augment them with the semantics of what they depict. Among their many potential uses they provide means for more efficient accessibility of graphical data going beyond the traditional use of textual alternative descriptions, such as natural language interfaces. However, no efficient way of authoring these graphics currently exists. This paper aims to bridge the gap between authoring graphics and enhancing them with semantic formal structures in the form of ontologies by introducing Semantic Annotator for Inkscape (SAI), an authoring tool that allows for seamless addition of semantics to an SVG file supported by a given upper ontology in RDF format. The traditional disjointed approach of authoring a vector image and editing its supporting ontology using independent software tools has thus been unified into a single workspace, improving the efficiency of authoring semantically-enhanced graphics. Evaluation of SAI has shown greatly improved annotation times of semantically-enhanced graphics that can be later used for efficient non-visual natural-language-based content retrieval.

Keywords

Blind/Low Vision, Software, Ontology, SVG, Web

Introduction

The World Wide Web has been designed with the ultimate goal of being accessible to everybody, regardless of age, location, device used, or whether the user has some disability. However, the presence of large amount of graphics on web sites can still present sometimes a huge barrier for specific user groups (Altmanninger and Wöß 378). One of the groups having the greatest difficulties accessing graphics is that of blind and visually impaired web users. Worldwide, 285 million people are estimated to be visually impaired, of whom 39 million are blind and 246 million have low vision (World Health Organization).

The easiest way of making graphics somehow accessible for visually impaired people is by describing them in text. On the web, this is generally done by means of the “alt” and “longdesc” attributes of HTML image elements ” (World Wide Web Consortium, *Web Content Accessibility Guidelines (WCAG) Overview*). This is a valid alternative for simple graphics whose meaning can be conveyed by means of short textual descriptions. However, in many cases an alternative textual description does not suffice for full understanding of the graphic. For instance, even though a textual and a diagrammatical representation might be informationally equivalent, they are generally not computationally equivalent, as verbal descriptions of diagrams easily overload short-term memory and do not provide any means of gaining an overview of data (L. M. Brown et al. 1).

Besides textual approaches, diagrams can be made accessible by means of spatial simplification e.g. by transforming a graph into a two-dimensional table in which each row and column represents a node and each cell the relationship between its two nodes (Blenkhorn and Evans, 22-25), sonification (representation of data through sound), tactile output, or using a hybrid approach. However, these methods suffer from a number of drawbacks that hinder their take up as an alternative to graphics on the Web. Sonification is not a suitable means of representing complex diagrams, whereas tactile and hybrid approaches require expensive equipment that moreover cannot be used by blind persons in an autonomous manner. Moreover, these methods are inherently incapable of displaying certain domains of graphical information, such as real-world photography. Recent methods to accessibility of Web graphics have thus shifted to audio- and speech-only approaches that do not require of any specific hardware or software, such as natural language interfaces.

Discussion

An innovative model for access to previously annotated pictures is the so-called communicative images (Plhák). A communicative image (e.g. Figure 1) is a two dimensional graphical object integrated with a dialogue interface (Natural Language Interface, NLI) and linked to an associated knowledge base in the form of ontologies which stores the semantics of the objects depicted. They consist of three data structures: (1) The graphic information itself, in vector or raster format; (2) identification of objects in the image, by exploiting the SVG (Scalable Vector Graphics) format to locate and give information of the objects within an image; and (3) semantic data associated to the graphic's objects, stored as metadata within the image in the form of ontologies encoded in OWL (Web Ontology Language).

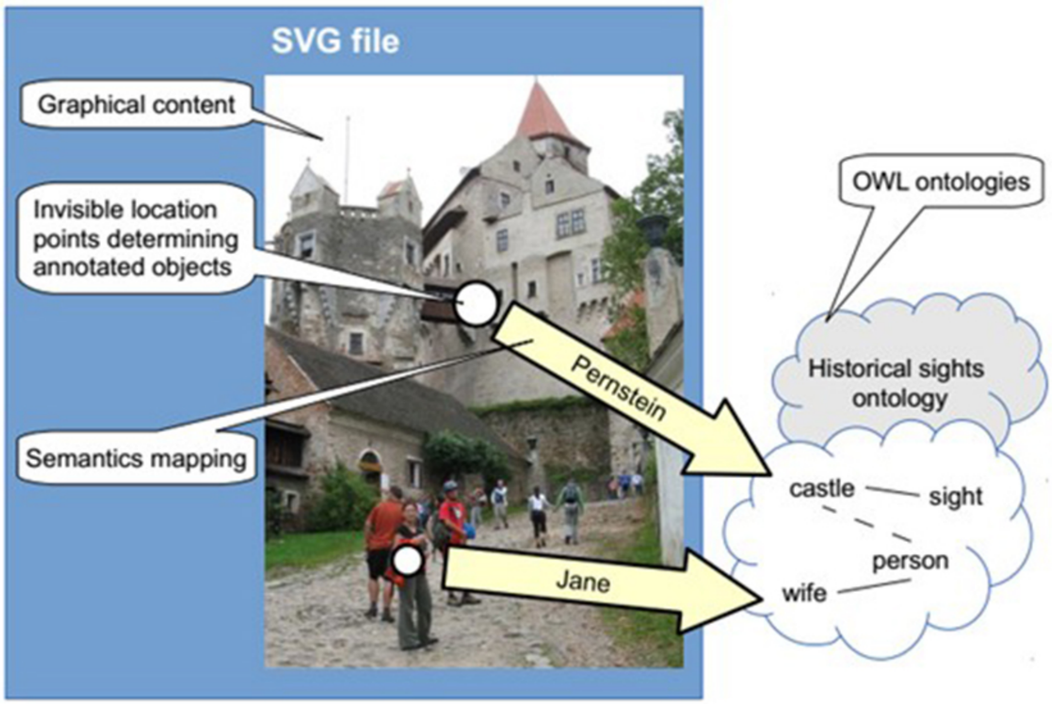

Fig. 1. Structure of a communicative image (Plhák)

Research shows that sound and speech interfaces to semantically-enhanced graphics are a fitting approach to display alternative accessible versions of real world images (Plhák 90), diagrams (A. Brown et al. 27) and charts (Ferres et al. 3). The serial nature of speech has an obvious disadvantage over tactile or sonification approaches which do not impose such a cognitive load on the reader's processing capabilities. Despite of this fact, blind readers much prefer speech feedback when reading charts and other means are preferred only rarely (Ferres et al. 3). Therefore, accessing the semantic information embedded in a communicative image by means of natural language allows blind users to autonomously infer the meaning of the graphic.

A required first step before annotating a communicative image is to segment it into regions. Several techniques for image segmentation are in use nowadays, such as edge detection methods, region growing methods, statistical methods or knowledge-based methods (Kopeček et al. 4-6). The resulting regions can then be recursively interpreted and given semantics by associating them to ontological instances (Kopeček et al. 8), as seen in Figure 1.

Within structured domains, using automatic object recognition techniques can greatly speed up semantic annotation. However, automatic object recognition techniques are currently far from being entirely reliable even with the use of large corpora of training data. Only in strictly controlled environments are these systems able to outperform the capabilities of the human visual system (Andreopoulos and Tsotsos 827).

Given the current imperfect nature of automatic annotation techniques, some sort of manual addition or edition of the semantic annotations present in a communicative image is commonly required. The manual annotation process is generally divided into two phases: (1) Authoring SVG – a variety of suitable tools are available for processing vector graphic elements, such as the proprietary Adobe Illustrator (Adobe Systems Incorporated), the free and open source Inskcape (The Inkscape Team) and GIMP (The GIMP Development Team). SVG elements either represent the graphic data themselves or interesting parts and objects in the bitmap image. In both cases, the created elements do not contain any semantical data and current tools do not allow the user to add them. (2) Annotating SVG elements – the most common tool for handling the semantical meaning is the free and open source application Protégé (Stanford Center for Biomedical Informatics Research). Protégé allows the user to efficiently process ontologies. However, it does not allow them to work directly with graphics and their constituent regions and semantic elements.

To the best of our knowledge, there are currently no available tools that allow users to process vector graphic images and annotate SVG elements enhanced with ontological data at the same time. Therefore, to tackle this issue the rest of this article presents an authoring tool for semantically-enhanced graphics. Semantic Annotator for Inkscape (SAI) provides an efficient bridge between the tools for processing vector graphic images and tools for assigning semantics to their elements. Its basic overview is given in the following section.

System Overview

The SAI prototype has been implemented as an extension of Inkscape, written in Python (https://www.python.org). Inkscape is one of the most popular graphics editors, working primarily with SVG files. In a typical usage workflow, the author may import an already existing SVG file, trace a raster image into its vector counterpart, or create a new SVG image from scratch. At any time during the authoring process, the user can choose to semantically annotate parts of the image by running the SAI extension from the Extensions menu or via a keyboard shortcut. This launches the SAI window (see Fig. 2). Therefore, complex graphics can be imported or created by the user in the same place as the semantic markup is added, and no further software is required, saving the user in both time and authoring complexity.

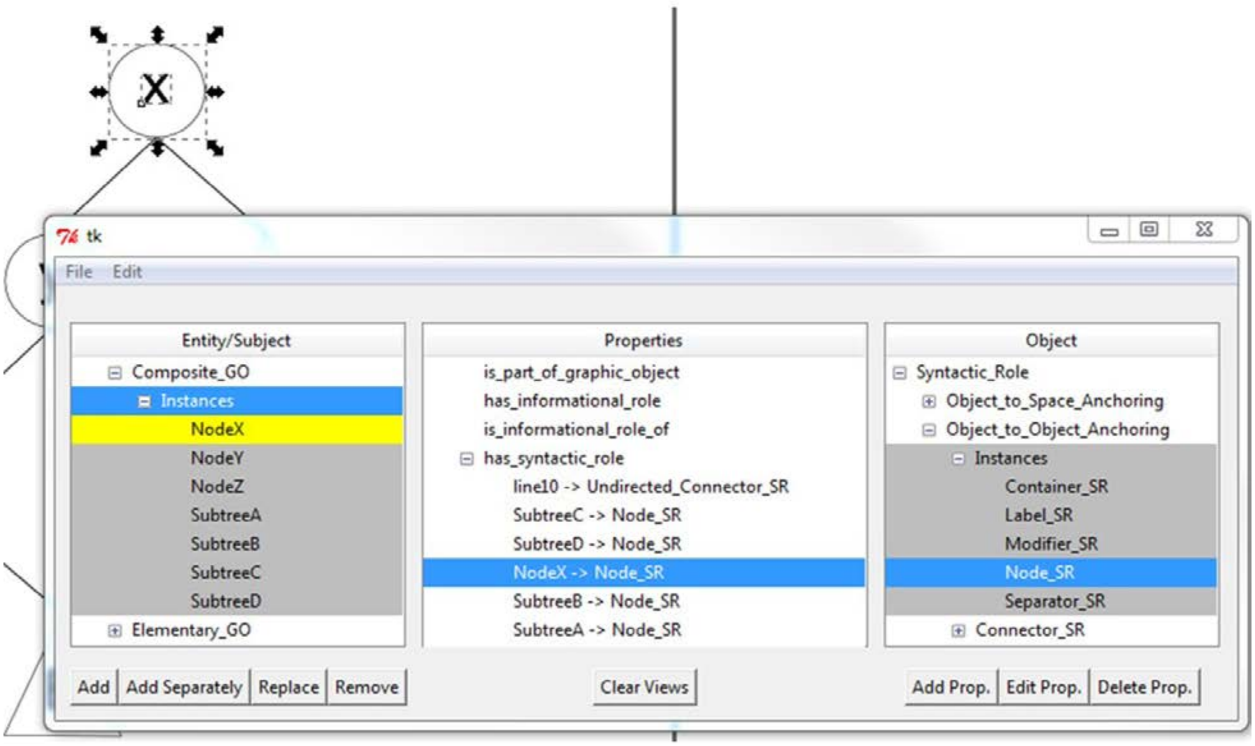

Fig. 2. The annotator GUI (foreground) in the process of annotating a red-black tree drawn in Inkscape (background).

Before any annotation of the graphic can take place, the user has to load a visualization ontology in RDF format by choosing the relevant option under the “File” menu. The extension tries to automatically infer the namespace used by the visualization ontology that semantically underpins the elements of the graphic and their relationships. If the namespace is not found, it prompts the user to manually input it.

SAI also checks whether the necessary datatype properties to link ontological entity instances to SVG elements (i.e. to semantically annotate the graphic) are present in the ontology. We associate the constituent SVG elements of an instance by means of datatype properties, which are added or edited automatically by our tool in a transparent manner to the user.

Any number of SVG elements present in an image (paths, shapes, text) may be selected by the author before running the annotator extension in order to relate them to certain ontological elements. A hierarchical list of entities and instances from the ontology appear on the left-hand tree view of the GUI. Instances can be distinguished from entities by their background color. At the same time, those instances to which some of the currently selected SVG elements belong are highlighted in the Entity/Subject tree view. The author can select entities and instances and then press one of the buttons underneath to perform the following functions:

Add: if an entity has been selected in the Entities tree view, a new named instance of said entity containing the chosen SVG elements will be created. The user will be prompted to add a meaningful name to this annotation. If an instance is chosen instead, the selected SVG elements will be added to it.

Add Separately: this functionality is similar to that of the “Add” button when used to create new instances of an entity. However, a new instance will be created for each of the selected SVG elements, instead of grouping them all together under the same instance. An automatic naming mechanism has been implemented in order to make the annotation faster. It takes into account the type of SVG element selected in order to give a meaningful and unique name to it.

Replace: substitutes all SVG elements present in the chosen instance for the currently selected SVG elements in Inkscape.

Remove: when one or more SVG elements have been selected in Inkscape, it will remove those elements from the chosen instance if possible. If no SVG elements have been selected, it allows the user to remove whole instances from the ontology.

Once a semantic annotation has been created by defining which SVG elements belong to an entity instance of the ontology, the user may define property assertions in which the annotation (instance) is involved. The system supports both object and datatype properties.

Properties and their occurrences within the ontology are also hierarchically displayed in the middle tree view of the GUI (Properties tree view). When a user double clicks on a property, the Entity/Subject (leftmost) tree view gets updated showing only those entities and instances that belong to the domain of the property, if defined. If the chosen property is an object property, the Entity/Object (rightmost) tree view will then display those entities and instances that belong to its range (i.e. the set of possible objects of the property triple). In case the user double clicks on a property assertion, the leftmost tree view will highlight the triple subject while the rightmost one will do the same for its object. This assists the user when choosing the elements of a triple to be added, edited or removed.

After selecting a property or property assertion in the Properties tree view, three operations can be carried out depending on which button below the Object tree view is clicked:

Add Property: lets the user add a new object property triple by selecting a subject (entity instance), property and object (entity instance). In case a datatype property is chosen, no object will appear in the rightmost tree view and the user will be prompted to add a value (e.g. a string or a number) to the triple instead. If the triple already exists, the user is informed and no triple is added.

Edit Property: lets the user select a property occurrence and modify its subject, object (value if the chosen property is a datatype property) or both by choosing a different instance in the Subject and/or Object tree views.

Delete Property: deletes the chosen property occurrence, regardless of the chosen subject and object.

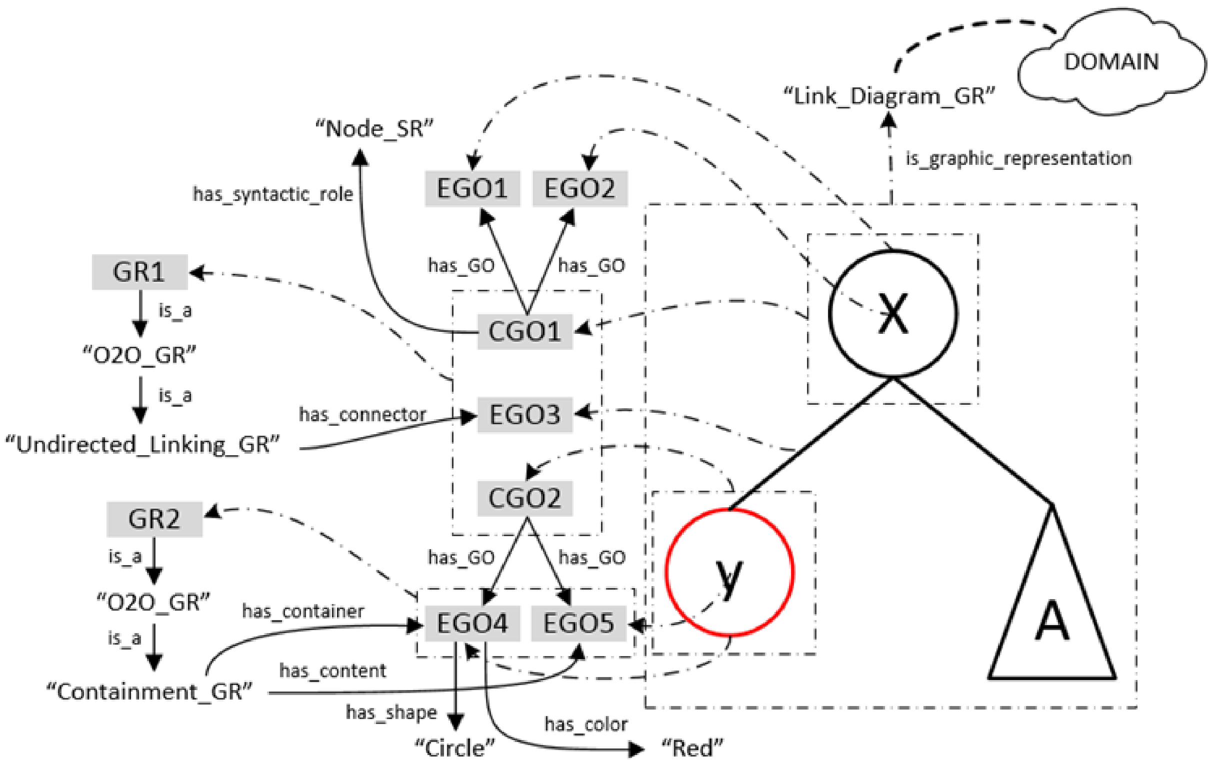

Fig. 3. Partial annotation of a red-black tree. EGO: Elementary Graphic Object; CGO: Composite Graphic Object; GR: Graphic Relation; O2O_GR: Object-to-Object GR.

Once the author has finished editing the image, the changes can be saved into the original ontology or a different one by choosing the appropriate option under the "File" menu item. All new instances and properties will be added into the selected ontology, and any edited/removed ones will be modified as well.

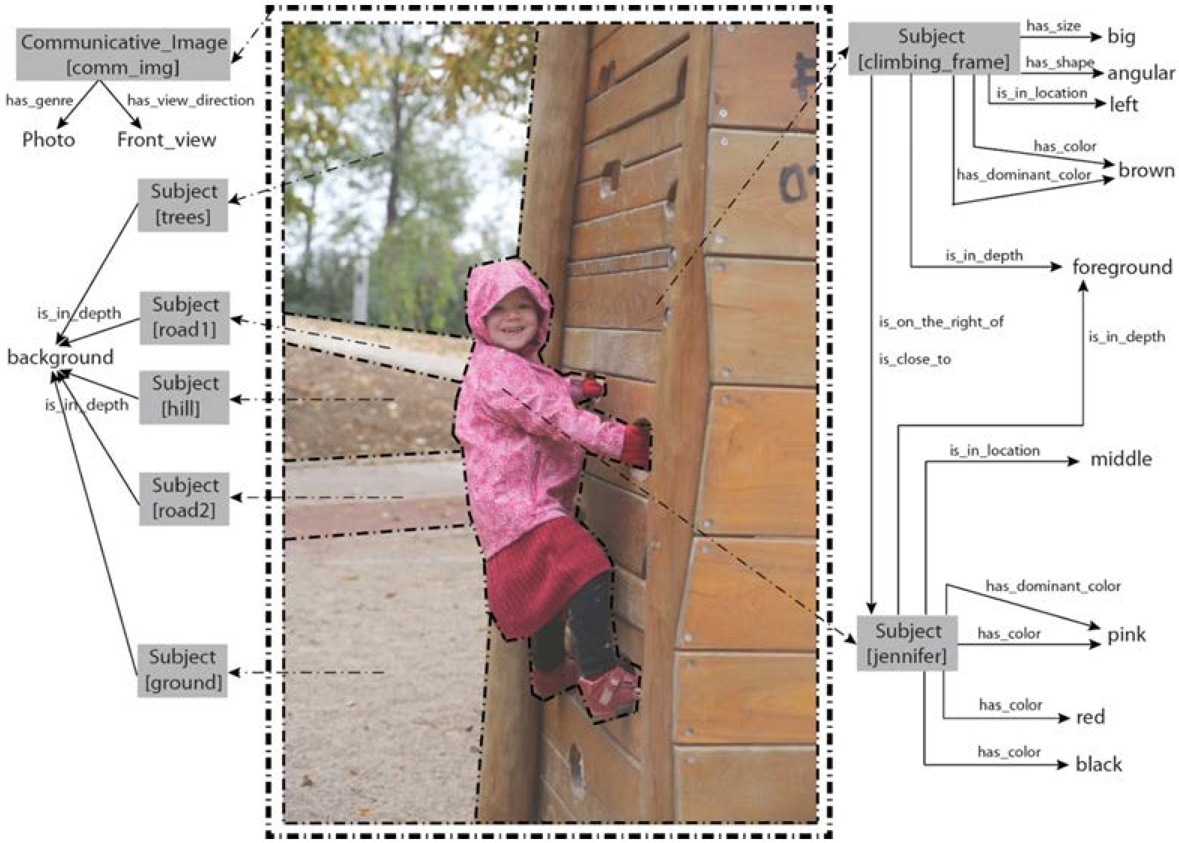

Fig. 4. Partial annotation of a personal photo using a graphical ontology.

Working Examples

The partial annotation of a red-black tree can be seen on Fig. 3. An upper ontology underpinning syntactical aspects of visualization (Murillo-Morales and Miesenberger, “Ontology-based Semantic Support to Improve Accessibility of Graphics”) is used. In the example, syntactical annotations about some of the elements that make up a tree diagram, such as nodes, connectors and linking relationships have been added. Domain-specific semantic annotations might be added later on by loading a specific domain ontology. Fully annotated diagrams may then be non-visually explored by means of natural language (Murillo-Morales and Miesenberger, “Non-Visually Performing Analytical Tasks on Statistical Charts” 343-345).

Fig. 4 shows the partially annotated photo from a personal album using a graphical ontology that restricts abstraction to the aspects that are suitable for dialogue-based exploration of graphical content. This graphical ontology contains general concepts common across different knowledge domains (Ošlejšek).

Benefits of SAI vs. traditional annotation workflow

Heuristic evaluation (Nielsen and Molich) of SAI with authors of semantically-enhanced SVG images both at the University of Linz and the Masaryk University Brno has shown a great reduction in annotation times, especially in the case of complex graphics. The rest of this section discusses some benefits of using SAI over traditional semantic annotation workflows.

Whereas using traditional tools such as Protégé requires naming each one of the entities in the ontology individually, SAI's automatic naming mechanism of ontological elements underpinning elementary graphic shapes along with the "Add separately" feature implemented in SAI saves the user substantial time in the naming process.

Highlight of selected SVG element in the user interface notably speeds up the authoring of property assertions by selecting their constituent elements in the graphic itself, instead of having to look them up by name (which might be non-descriptive). For example, a user might choose to add a syntactic role to all bars in a bar chart by selecting them in Inkscape and then launching SAI, where all entities they belong to will be highlighted and may be chosen to create new property triples.

SAI saves the author substantial time when adding several instances of the same entity to the supporting ontology. It remembers the last class chosen when adding a new individual, so its parent Entity is automatically selected in the GUI when relaunching the extension. This way, the author simply needs to click on the 'Add' button after selecting the SVG elements that will make up the new individual. This is especially useful in knowledge representation graphics, as the same graphic relations may apply to many different elements e.g. labelling relations on a map with one label per geographical landmark.

For every instance in the ontology, their approximate coordinates within the SVG canvas are computed by taking into account the coordinates of its constituents SVG elements and included in the ontology as datatype properties. Authors can make use of the transform SVG attribute (automatically applied by Inskcape) to scale, rotate or translate elements, and the resulting end coordinates on the SVG canvas are calculated by using transformation matrices (World Wide Web Consortium, *Coordinate Systems, Transformations and Units*)

Conclusions

The question of adding semantics to graphics has received much research attention in recent years. SAI is a first prototype of an authoring tool that bridges the existing gap between authoring graphical information and adding semantic underpinnings to it. Authors can avoid using complex tools for processing ontologies, and annotate images in a more convenient way directly in the vector graphics editor.

Currently, SAI supports loading RDF ontologies from a single file and is oriented to processing upper ontologies. In the future, we plan to support loading more ontologies from multiple files and extend the functionality to support processing domain ontologies, e.g. for touristic sights, personal relationships, specific tree structures, etc. Moreover, we would like to include automatic processing of metadata in the EXIF data (date, GPS coordinates) as well as automatic image segmentation techniques and other authoring time-saving techniques.

Works Cited

Adobe Systems Incorporated. *Adobe Illustrator CC.* www.adobe.com/products/illustrator.html.

Altmanninger, Kerstin, and Wolfram Wöß. "Accessible Graphics in Web Applications: Dynamic Generation, Analysis and Verification." *Computers helping people with special needs: 11th international conference, ICCHP 2008, Linz, Austria, July 9-11, 2008 : proceedings*, edited by Klaus Miesenberger et al., Springer, 2008, pp. 378–85. Lecture notes in computer science, 0302-9743 5105.

Andreopoulos, Alexander, and John K. Tsotsos. "50 Years of object recognition: Directions forward." *Computer Vision and Image Understanding*, vol. 117, no. 8, 2013, pp. 827–91. doi:10.1016/j.cviu.2013.04.005.

Blenkhorn, P., and D. G. Evans. "Using speech and touch to enable blind people to access schematic diagrams." *Journal of Network and Computer Applications*, vol. 21, no. 1, 1998, pp. 17–29.

Brown, Andy, et al. "Making Graph-Based Diagrams Work in Sound: The Role of Annotation." *Human-Computer Interaction*, vol. 28, no. 3, 2013, pp. 193–221. doi:10.1080/07370024.2012.697010.

Brown, L. M., et al. "Design guidelines for audio presentation of graphs and tables." *Proceedings of the 9th International Conference on Auditory Display*, July 6 - 9, 2003, Boston University, Boston, MA, USA. Boston, MA Boston University, 2003.

Ferres, Leo, et al. "Evaluating a Tool for Improving Accessibility to Charts and Graphs." *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 20, no. 5, 2013, p. 28.

Kopeček, Ivan, et al. "Detection and Annotation of Graphical Objects in Raster Images within the GATE Project." *Proceedings of the 2009 International Conference on Internet Computing: ICOMP 2009*, edited by H. R. Arabnia and Victor A. Clincy, CSREA Press, 2009, pp. 285–90.

Kopeček, Ivan, et al. "Integrating Dialogue Systems with Images." *Text, speech and dialogue: 15th international conference, TSD 2012, Brno, Czech Republic, Sep. 3-7 2012 : proceedings,* edited by Petr Sojka, vol. 7499, Springer, 2012, pp. 632–39. LNCS sublibrary. SL 7, Artificial intelligence 7499.

Murillo-Morales, Tomas, and Klaus Miesenberger. "Ontology-based Semantic Support to Improve Accessibility of Graphics." *Studies in health technology and informatics,* vol. 217, 2015, pp. 255–60.

---. "Non-Visually Performing Analytical Tasks on Statistical Charts." *Harnessing the Power of Technology to Improve Lives,* edited by P. Cudd and L. de Witte, vol. 242, IOS Press, 2017, pp. 339–46.

Nielsen, Jakob, and Rolf Molich. "Heuristic evaluation of user interfaces." *CHI '90 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* edited by Jane Carrasco Chew and John Whiteside, ACM, 1990, pp. 249–56.

Ošlejšek, Radek. "Annotation of Pictures by Means of Graphical Ontologies." *International Conference on Internet Computing: ICOMP,* CSREA Press, 2009, pp. 296–300.

Plhák, Jaromír. *Dialogue-based Exploration of Graphics for Users with a Visual Disability.* Masaryk University, Faculty of Informatics, 2014.

Stanford Center for Biomedical Informatics Research. *Protégé.* protege.stanford.edu/

The GIMP Development Team. *GNU Image Manipulation Program: GIMP.* www.gimp.org/.

The Inkscape Team. *Inkscape*. inkscape.org/.

World Health Organization. *Visual Impairment and Blindness Fact Sheet.* 2014, www.who.int/mediacentre/factsheets/fs282/en/. Accessed 22 May 2017.

World Wide Web Consortium. *Web Content Accessibility Guidelines (WCAG) Overview*. 2005, www.w3.org/WAI/intro/wcag. Accessed 22 May 2017.

---. *Coordinate Systems, Transformations and Units.* 2011, www.w3.org/TR/SVG/coords.html Accessed 22 May 2017.