Creating, Interpreting, and Repurposing Visual Messages

Timothy Lebo[1,2] and Deborah McGuinness[1]

[1] Rensselaer Polytechnic Institute, Troy, NY USA
[2] Air Force Research Laboratory, Information Directorate, Rome, NY USA

Abstract. The World Wide Web is a vast, diverse, and dynamic ecosystem of content authored and consumed with innumerable frequency. Although content may have an original purpose, it can and will be repurposed in new and unexpected ways. Research in the Semantic Web has led to the pragmatic adoption of several technologies that establish the groundwork for increased machine-to-machine interoperability. Nevertheless, the human factor remains – and is paramount – in both the original and semantic webs, and so is the task of transforming mechanistic content for human consumption and comprehension. Unfortunately, traditional transformation methods produce isolated artifacts incapable of inspection, elaboration, and extension. This paper presents a methodology for creating accountable visual artifacts – visual messages that preserve associations to their source content, allow traceability to their creation, and provide guidance for proper interpretation. The system is described by following two RDF triples through the originating visualization process, into application-specific formats, and back to an RDF-based visual transcription. We conclude by describing future work that will further increase machine accessibility, while maintaining human approachability, of visual messages on the web.

1 Introduction

The World Wide Web allows people to publish content for others to consume. Content may reach its intended audience or an unintended audience, and it can be used as intended or not as intended. The effectiveness of some content's form is dependent on the intents of both its author and audience. If the intent of the audience matches that of the author, it is likely that the content is effective without modification. However, modifications may be advantageous – or even required – if the author made improper decisions about the content's form. When used by an unintended audience, the content's form is likely unsatisfactory, since the author had not considered or appreciated these contexts when establishing implicit assumptions about the intended audience. Fig. 1 illustrates the likelihood of repurposing given the types of audience, use, and effectiveness for a task. As audiences grow, time passes, and intents change, content will likely require manipulation to satisfy current needs.

Audience	Use type	Effectiveness	Re-purposing
Intended	Intended	more effective	not likely
		less effective	possible
	Unintended	more effective	possible
		less effective	likely
Unintended	Intended	more effective	not likely
		less effective	possible
	Unintended	more effective	likely
		less effective	probable
none	none	none	none

Fig. 1. Likelihood of repurposing given type of audience, use, and effectiveness for task.

For example, a recent blog entry[3] describes the use of a new recommendation system. The author begins with a link to background information and offers a raster image of an histogram showing the number of acceptances that Citeulike users performed using a new article recommender on each day from 21 Sept 2009 to 2 Nov 2009. The histogram is reproduced to the right of Fig. 2. Fortunately, we found this image through the blog entry and not in isolation, so we have a few extra clues to help us correctly interpret the image. The title and axes are also reasonably labeled within the image, so there is a chance that an audience could decipher the intended message if they stumbled upon the isolated image. In this situation, the audience's intent matches that of the author; we want to know the number of acceptances the recommendation system earned on each day. Still, an human audience tend to make a lot of assumptions to reach a correct interpretation, and a machine would be at a loss. As we scroll down the blog entry, we see that another histogram is offered in the same rastered format. This one shows the number of *rejections* that the same recommendation system earned in the same time period. The pair of histograms now falls into the "less effective" category, because the author's intent for us to compare acceptances to rejections becomes awkward at best given the spatially disparate content.

So far, the example considers more- and less-effective forms of content for *intended* use by an *intended* audience. These two situations are *least likely* to require content manipulation, since repurposing is not required. When the content is used by an unintended audience, or the content is used by anyone in unintended ways, the likelihood of content manipulation for repurposing increases. This is simply because the author had not considered these audiences or uses, and thus could not reasonably design or prepare for these situations. But just because an author does not intend for particular audiences or uses, does not mean that the author wants to exclude them. On the contrary, the intent of the blog entry author is to engender interest in the recommendation system. However, the rastered form of the content remains uninformative for any interested reader with insights or inquiries. Interested audiences want to know more. They may reasonably ask how many users exist, how many users are "active", how many used the new recommender, or how the acceptances and rejections tally

[3] *Data from CiteULikes new article recommender*, http://blog.citeulike.org/?p=136

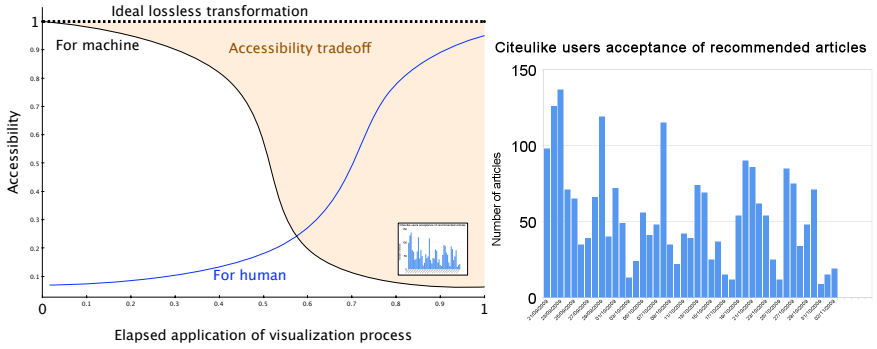

Fig. 2. Raster image linked from a blog entry describing a new article recommender.

by day of the week or by hour of the day. The left half of Fig. 2 illustrates the accessibility tradeoff that prevents machines from supporting an audience's investigation or elaboration of some visual message.

Although this histogram example is taken from the traditional web and not the semantic web, the mistreatments of visual messages in both are essentially the same. The problem is two-fold. First, interested audiences may implore the content for how it was created to better understand the context of its current message. Second, interested audiences may implore the content for information related to its current visual message. As described earlier, current methodologies sever the relevant associations, leaving the visual artifact isolated and unable to satiate these needs. If the only concrete form of the web that audiences experience is dissociated from the web's true mechanistic forms, then audiences depend on a limited few to craft answers to satisfy their interests. The current developer-in-the-loop approach cannot scale with the size of the web and the size of its audience.

2 Related Work

Karger and Schraefel [7] review a cross-section of efforts to visualize semantic web data. They criticize the so-called "graph-based" visualizations and advocate a return to traditional "un-graph" interface approaches. They advocate the continued use of traditional user interfaces which would simply be populated with RDF content instead of heterogeneous formats. However, this approach reinforces the dissociation between the visual forms in the interface and the originating content, relying on interface developers to procedurally maintain these connections and forcing audiences to assume or remember each interpretation.

To address the two problems with visual content generation described in the previous section, two aspects of the same solution should be considered. To answer questions about how visual content came to be, the transformation from content to visual form should be prescribed *a priori* and logged upon execution.

To answer related questions about the content within a visual message, the message should preserve associations to the content that it represents.

To realize semantic visual objects, the transformation from content to visual form should be prescribed. During the 1990's, the AI and visualization communities actively investigated automated visual creation systems [9], but this pursuit waned by the turn of the century. Graph Style Sheets were an early effort that led to the Fresnel RDF Display Language [8]. Although Fresnel is closer to a declarative visual strategy than most, it has been criticized for only being a processor language. Regardless, it has not gained in popularity, probably due to its informal formulation and limited availability of documentation, examples, capabilities, and robust implementations. It does not leverage any established semantics to perform visual encoding decisions, and is geared to produce CSS-like visuals. By design, Fresnel is also indifferent to the distinction between visual connection and visual containment, leaving an all-or-none interpretation decision to visual designers consuming its output.

To realize semantic visual objects, the visual artifact should preserve the content it represents. RDFa allows authors to augment the visual data in XHTML with machine-readable content [1]. Both the machine-readable and the human-readable text comes from the exact same location on the page, ensuring that both humans and machines have the most up-to-date data. Although this duality facilitates recovery of the source content, the convention only applies to HTML. Further, RDFa does not provide traceability to how the page was created or from which sources the content originated, treating visual elements as a second class entity. There is also no guarantee that the human audience will be able to infer the underlying content based solely on the rendering of the XHTML. GRDDL strives to be a general-purpose RDF extractor for formats that are not natively RDF-based. Because of this generality, an RDFa parser could work as a special case within the GRDDL framework. Although GRDDL's generality could also be used as a framework to recover content from a visual artifact, we are not aware of work that demonstrates this. GRDDL also assumes that those adopting it have some control over the non-RDF formats, which is not the case with established graphical formats of popular applications.

3 A Terminology for Semantic Visual Objects

To understand visualization more deeply, many have proposed decompositions for a generalized visualization system [3, 5]. However, they use system-level approaches instead of focusing on the content-level structures upon which the systems operate. To avoid confusions that arise from the many uses of the term *visualization*, we distinguish three related notions: *visual strategy*, *visualization process*, and*visual form*. Each of these follows the previous in a progression from physical data to psychological understanding.

First, **visual strategy** is the prescribed plan of action or policy designed to achieve the overall aim of communicating some subset of content to an intended human audience. Example visual strategies could be to *show all coffee shops*

open past 8pm as a list of names; *show all of Jane's friends as squares colored by gender*; or *show all blogs mentioning* iPad*, making more recent blogs less transparent and adding a reddish tint to those mentioning it before 27 January, 2010*. Unfortunately, visual strategies are often defined procedurally and coupled with the implementation of the visualization process.

A **visualization process** is the execution of a visual strategy for some available content. An invocation of the visualization process results in a single *visual artifact* that comprises a variety of *visual forms*[4]. A visual artifact is a file that could be in a variety of graphical formats, such as PDF, SVG, HTML, or PNG.

Visual forms are described using *visual attributes* such as shape, position, and color [2]. If the coffee shop visual strategy above was applied during a visualization process, and HTML was the chosen graphical format, each `<div>` containing the name of a coffee shop would be a visual form and its visual attributes would be described using CSS styles. If SVG had been chosen for the graphical format, each `<g>` element containing a shop's name would be a visual form, and its visual attributes would be described using SVG's drawing commands. *Visual relations* among visual forms are created based on the forms' visual attributes; containment, alignment, and connection are all types of visual relations. For example, visual forms depicting the coffee shop name, walking distance, and closing time could be contained by an enclosing visual form colored to distinguish it from neighboring visual forms depicting other coffee shops. The left borders of all visual forms in the coffee shop example could also be aligned to ease readability. A visual form's appearance above another is a different type of visual relation, and could be used to indicate that a coffee shop is closing later than another. *Visual constraints* restrict the values of visual attributes and relations for certain visual forms, and *visual constructs* are sets of visual constraints. An example visual constraint is *the vertical position of a coffee shop open later should be greater than the vertical positions of those that close earlier*. Other examples of visual constructs include bar charts, scatter plots, contour maps, tree maps, heat maps, parallel coordinates, cone trees, and atoms [6].

4 System architecture

4.1 Pull-oriented versus Shove-oriented Visual Strategies

A common pattern when creating a visual message is to perform a query and place the content into a visual construct. For example, we could query for *all non-smoking coffee shops within a 15 minute walk* and render this as a list or as a map. This "pull-oriented" approach places a burden on the visual designer to determine what structures actually exist in the source content, which are important for the audience, what will be queried, and how each will be shown. The visual designer makes assumptions to handle each of these burdens. Although burdens and assumptions were manageable when we controlled the development of our

[4] The relationship between a visual artifact and a visual form is much like the relationship between a named graph and a set of RDF triples [4].

data source, the semantic web's continuous evolution sacrifices our control in exchange for a wealth of additional information. This motivates new approaches for crafting visual messages, since approaching an entire unfamiliar data set is now a very commonplace situation.

We believe the adaptability challenges imposed by an evolving semantic web suggest the visual strategy should be made explicit. Explicit assumptions embody the visual designer's understanding of what the audience needs, they enable inspection to address a variety of inquiries about a particular visual message, and they can be reused for subsequent applications. Instead of selecting specific subsets of content, a "shove-oriented" visual strategy considers *all* content elements for inclusion in the resulting visual artifact, and applies the explicit assumptions to make visual encoding decisions. In the case where no content elements are excluded, an "identity transform" will result in what has been called the Big Fat Graph [7]. This assumption-free visual strategy provides an informative first approach to an unfamiliar data set and allows the visual designer to craft the first assumption for the developing visual strategy. The exclusion of some content is an important early step, and results in the assumption that the audience either knows this content already, can infer it, or does not need to know it for the task at hand. Additional assumptions are made about the audience when controlling visual attributes and relations among particular visual forms.

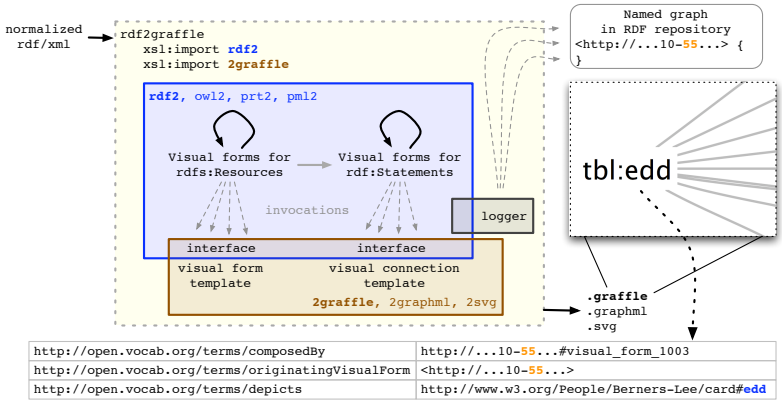

Fig. 3. Visual strategies (blue) are coupled to target graphical formats (brown) through a visual form interface and visual connection interface. The pairing (yellow) of a visual strategy with a target graphical format is controlled by changing which are in scope upon execution of the visualization process. Essential triples linking the application-specific format to the remaining linked data are encoded using application-specific metadata constructs.

4.2 Current implementation

We have implemented a shove-oriented visualization system using XSLT 2.0. The system inputs normalized RDF/XML, which does not use nesting, uses `rdf:Description`s instead of the typing shorthand, and names all blank nodes.

Encoding decisions create visual forms to depict content elements. First, relevant information about the triple's subject, predicate, and object is gathered to determine relevant classifications. These classifications are then used to assign the values for each visual attribute in the visual connection template. After logging justification for the determined values, the visual connection template is invoked to produce an appropriate fragment of the target graphical format. When decisions are made to omit a visual form depicting a content element, the justification is logged, but the graphical format template is not invoked. After all inputs, a single visual artifact in the specified graphical format is produced.

Interfaces to create visual forms and visual connections are used to decouple the visual strategy from any specific graphical format. The visual form interface accepts descriptions for *spatial extent* (x, y, height, width, shape, rotation), *border stroke* (thickness, color, stipple), and *label* (alignment and padding). The visual connection interface accepts the source and target visual forms to connect, along with descriptions for the connection's *stroke* (width, stipple, color, shadow), *label* (color), and *end styles* (none, arrow, X). The most mature graphical format templates that we have developed are for OmniGraffle 5. Additional graphical format templates should be created for Graphviz, SVG, HTML, and GraphML.

4.3 Provenance of Visualization Processing

For each invocation of a visualization process, a named graph is created on a visualization endpoint to describe the encoding decisions it used to produce the visual artifact. As encoding decisions are logged during the visualization process, their RDF description is published to the named graph. This includes the name of the `xsl:template` handler that made the encoding decision, the classifications it used, the visual attribute's assigned value, and a timestamp.

The names of all `Visual form`s and the rdfs:Resources that they depict are also described in the named graph. This provides a minimal description of the resulting visual artifact and enables queries for depictions of a given rdfs:Resource. Since graphical formats are designed to describe what *to draw* instead of what *not* to draw, they cannot be used to record decisions to omit a visual form for a content element. Since these decisions are described in the named graph using `Null visual form`s, the visualization endpoint provides supplemental information that the visual artifact cannot convey.

This additional information enables the visual artifact to answer a new class of inquiries. If an audience familiar with the depicted content notices an omission and wants to know why it is not shown, the system can firmly provide one of three answers. Using the example illustrated in Fig. 4, Tim Berners-Lee may ask why the visual is not showing that he knows Ivan Herman, as it

is only showing that he knows Edd Dumbill. The visual artifact's provenance can be queried on the visualization endpoint for any visual forms depicting `http://www.ivan-herman.net/foaf.rdf#me`. If no results are found, then Ivan is not being depicted because the content was not available and thus not considered during the visualization process (this is the case in Fig. 4). If a `Null visual form` depicting Ivan's URI is found, then he is not being depicted because an encoding decision was made to omit him from the visual artifact. The stored justification for the omission can be provided. For example, the visual strategy may have incorporated an assumption that only Tim's friends from the UK should be depicted. Finally, if a non-null `Visual form` depicting Ivan's URI is found, then it should be depicted and may have been deleted from the visual artifact or may be currently out of view.

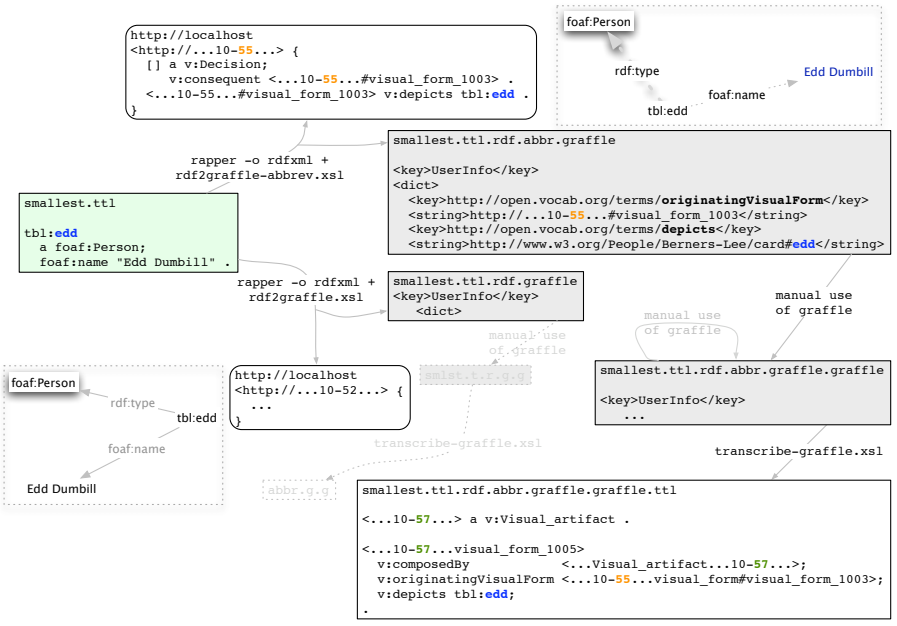

Fig. 4. Lineage of data transformations for the visualization of two FOAF triples. Two visual artifact chains are created by two visualization processes, each applying a different visual strategy to `smallest.ttl`. Encoding decisions for the visual artifact and its visual forms are stored in a named graph on the visualization endpoint `http://localhost`. In the gray boxes, application-specific metadata mechanisms are used to encode the few essential triples to associate the visual form to the remaining linked data. A GRDDL-like process grows the visual artifact chain by creating a visual transcription using terms from the visual forms ontology. Visual forms maintain links to their originating visual form and their referent (e.g., `tbl:edd`). The node-link diagrams embraced by dotted borders illustrate the different visual artifacts resulting from the different visual strategies.

4.4 Extracting a Visual Form's Referent

If an unintended audience were to inspect a rendering of the visual artifact, they would see shapes with certain colors at certain positions – they would not see Universities of New York State, and they would not see coffee shops within a 15 minute walk. Guidance for correct interpretation should be provided so that the audience sees Universities instead of squares. Current approaches rely on either an informed audience or provide a loosely associated narrative to "share the secret decoder ring". An approach to semantic visual objects should account for the popularity of existing graphical formats. Fortunately, most formats recognize the need for extensibility and permit user-defined attribute-value pairs. This construct is used to encode a minimal set of triples on each format-specific visual form to preserve their association to the visualization process that created them (using `v:originatingVisualArtifact`) and the rdfs:Resource they depict (using `v:depicts`).

The architecture and implementation just described also creates visual artifacts that can be published, copied, and modified at will. However, the content depicted by visual forms can be inspected, queried, and elaborated in three distinct ways. First, the visualization endpoint can be queried for visual artifacts that were created and what rdfs:Resources they depict. When searching for information about a particular resource, a list of all visual artifacts depicting the resource can be provided. This explicit knowledge answers more direct questions than image processing and page context heuristics used in image search engines. Second, if an audience has a visual artifact, but is not aware of the visualization endpoint and its supplemental linked descriptions, the user can use the application's existing interface to obtain the URI of the resource that the visual form depicts. This provides an entry point into the remaining semantic web using the variety of existing navigation and search tools. The user can also obtain the URI of the originating visual form, which leads back to the visualization endpoint using linked data principles. Third, a visual transcription of the visual artifact can be created using a transformation created for its graphical format. The visual transcription is an RDF description of the visual artifact and represents a snapshot in time of the visual artifact, using a common vocabulary across all graphics formats. This allows, for some set of obtained visual artifacts, autonomous extraction of the visual content and their referents for query and analysis without dependence on the visualization endpoint.

5 Future Work

Our current shove-oriented visualization system uses handlers implemented in a functional language. Upon noticing the pattern described in section 4.2, we intend to define a number of declarative handlers to facilitate greater transparency. With this approach, a visual strategy would be written as an OWL ontology comprising intersections among Depictors and Visual forms. The visualization system would use the ontology to create the visual artifact, and the same ontology could be used to interpret the visual transcriptions described

in Sec. 4.4. Approaches from the automated visual creation literature should be further reviewed and compared to both of these approaches. Although visual strategies have been created for instance data, OWL, and a rules language, additional content types would clarify the current design and broaden its applicability. Graphical format templates beyond OmniGraffle 5, such as Graphvis, GraphML, or SVG should be added to develop the visual form interfaces and extend the types of audiences. Adding graphical format templates will require a corresponding visual transcriber. The URIs created for visual artifacts, forms, and encoding decisions were prototypical, but their structure should be designed and appropriately deployed as well-principled Linked Data. Similarly, the encoding decision representation should be reconciled with generally accepted provenance markup languages such as Proof Markup Language or Open Provenance Model. While most graphical formats do not permit a combination of visual containment and visual connection, hybrid uses may yield better visual strategies. A visual containment interface and template should be added, with an investigation into how it can be combined with the containment interface and template.

References

1. B. Adida, M. Birbeck, S. McCarron, and S. Pemberton. Rdfa in xhtml: Syntax and processing, October 2008.
2. M. Bugajska. *Spatial Visualization of Abstract Information: A Classification Model for Visual Spatial Design Guidelines in the Digital Domain*. PhD thesis, Swiss Federal Institute of Technology Zurich, 2003.
3. S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1 edition, February 1999.
4. J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM.
5. G. K. chu Ng. Interactive visualisation techniques for ontology development, 2000.
6. B. M. K. Samp, C. Basca and S. Decker. Atom interface - a novel interface for exploring and browsing semantic space. In *Semantic Web User Interaction at CHI 2008: Exploring HCI Challenges*, 2008.
7. m.c. schraefel and D. Karger. The pathetic fallacy of rdf. In *International Workshop on the Semantic Web and User Interaction (SWUI) 2006*, 2006.
8. E. Pietriga, C. Bizer, D. Karger, and R. Lee. Fresnel: A browser-independent presentation vocabulary for rdf. In I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. M. Aroyo, editors, *The Semantic Web - ISWC 2006*, volume 4273, chapter 12, pages 158–171. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
9. M. X. Zhou. *Automated Generation of Visual Discourse*. PhD thesis, Columbia University, New York, NY, USA, 1999.