Combining Generality and Specificity in Generating Hypermedia Interfaces for Semantically Annotated Repositories

Lloyd Rutledge
Lloyd.Rutledge@cwi.nl
CWI
PO Box 94079
NL-1090 GB Amsterdam, the Netherlands

Geert-Jan Houben **Flavius Frasincar**
G.J.Houben@tue.nl F.Frasincar@tue.nl
Eindhoven University of Technology
PO Box 513
NL-5600 MB Eindhoven, the Netherlands

Abstract
Hera and Topia are separate research implementations for generating hypermedia presentations from media repositories annotated with Semantic Web technologies. They potentially define two cooperating components of a larger system. Topia creates a general, high-level outline or story board, while Hera can be used to fill in the details and make the smaller-scale components have a rich and engaging presentation. This paper explores combining Topia and Hera, describing along the way the resulting issues for providing hypermedia interfaces to semantic annotations.

Categories & Subject Descriptors
H.1 [**Information Systems**]: Models and principles; H.5.4, H.5.1 [**Information Interfaces and Presentation (e.g., HCI)** Hypertext/Hypermedia - Architectures, Navigation; Multimedia Information Systems - Hypertext navigation and maps, Evaluation/methodology; I.2.4 [**Artificial Intelligence**]: Knowledge representation formalisms and methods; D.2.2 [**Software Engineering**]: Design tools and techniques; I.7.2 [**Document and Text Processing**]: Document Preparation - Hypertext/hypermedia, Markup languages, Multi/mixed media, Standards

General Terms
Algorithms, Documentation, Design, Experimentation, Human Factors, Management, Standardization, Languages, Theory

Keywords
WIS, User Interaction, RDF(S), Semantic Web

Introduction

In the Web, the use of hypermedia presentations is a useful way to convey information. This is especially true in applications that try to give access to large collections of data. For these applications the use of hypermedia presentations is an effective way to convey the semantics of the content to the user. There are numerous interesting approaches that aim at generating hypermedia presentations for the output of Web applications. Many of these approaches use Semantic Web technology as a way to have an effective extensible and flexible format to describe the properties of the content to be presented in the application.

In this paper we show how the combination of two such systems, Topia [5] and Hera [8], and their accompanying tool sets can provide an environment for generating hypermedia presentations over Web content from semantic annotations over media repositories. We can join Hera and Topia as two cooperating components in a larger system. This is possible because Topia intentionally leaves presentation details for other components of a broader system to process since Topia is specifically a large-scale presentation structurer. In this context Hera, with its more controlled presentation generation, can provide domain-specific modules that process Topia output or communicate with Topia to state requirements on the output. This means that in this combination, Topia creates a general, high-level outline or story board, while Hera can be used to fill in the details and make the smaller-scale components have a rich and engaging presentation.

Besides the functional match, the Hera and Topia approaches also match in terms of the formats, languages and tools used. Both use an architecture with RDF(S), SeRQL, XSLT and Java. Topia performs data transformations based on an algorithm that directly works with instances though its combination of XSLT and SeRQL. Hera's data transformations are specified at schema level using the following overlay models: Conceptual Model (CM), Application Model (AM) and the Presentation Model (PM). These transformations are encoded in XSLT and SeRQL. Topia's semantic graph corresponds with Hera's CM instance. Topia's structured progression corresponds with Hera's AM instance. Topia's transformations correspond with Hera's transformations. Topia and Hera illustrate their possibilities on the same use case as both process the same RDF-encoded repository of annotated media items about the Rijksmuseum Amsterdam collection.

Hera

The Hera methodology [8] is a model-driven methodology for designing Web information systems (WIS). In response to a user query, a WIS gathers multimedia data, possibly

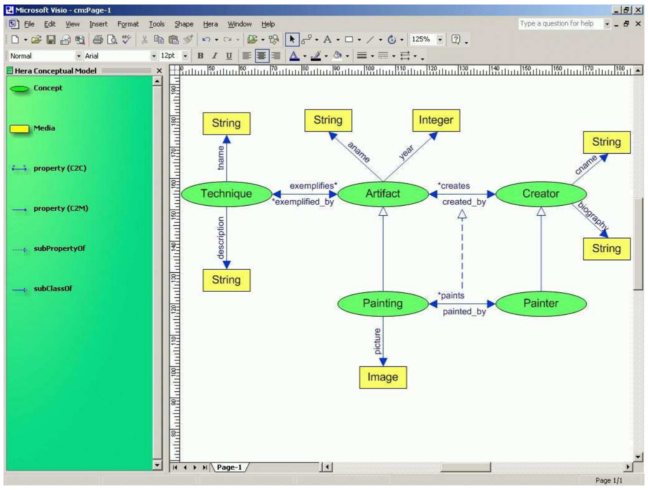

Figure 1: Conceptual model

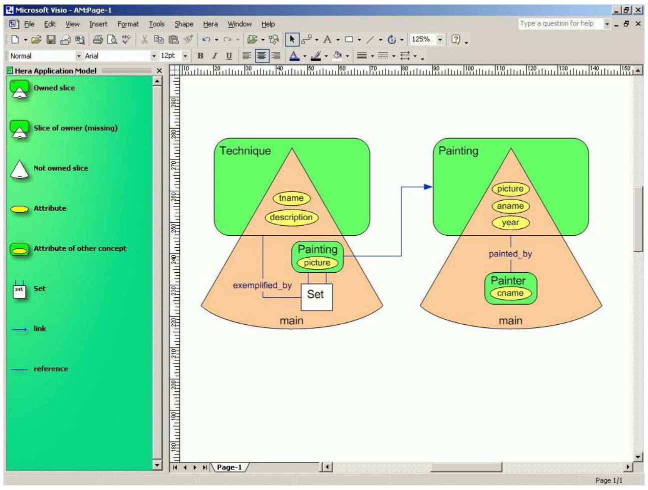

Figure 2: Application model

from heterogeneous sources, and produces a meaningful hypermedia (Web) presentation for the retrieved data. The Hera methodology automates this process by providing high-level model-based abstractions that drive (semi-)automatic presentation generation. Moreover, Hera enables presentation adaptation based on user preferences and device capabilities, which means that the presentation generation takes into account issues like types of platforms used, such as desktop, PDA or WAP phone [2].

Based on the principle of separation of concerns and for the sake of interoperability, several models have been distinguished. Because these models are considered Web metadata descriptions that specify different aspects of a WIS, the choice was made to use the Web metadata language RDF(S) to represent all models and their instances. This choice is also justified by the RDF(S) extensibility and flexibility properties that enable to extend the language with model specific primitives to achieve the desired power of expression. As RDF(S) doesn't impose a strict data typing mechanism it proved to be very useful in dealing with semistructured (Web) data.

The Hera tool set implements this methodology by offering software for the automatic generation of hypermedia based on the different Hera models. In order to facilitate the building (and visualizing) of these models, several Visio solutions were implemented. Such solution is composed of a stencil that will display all the model shapes, a drawing template, and a load/export feature providing the RDF(S) serialization of Hera models. Figure 1 depicts (in the CM Builder) a part of the CM for our Rijksmuseum example application, while Figure 2 illustrates the corresponding AM (in the AM Builder).

The CM describes the application domain in terms of concepts and concept relationships. A concept has attributes, i.e. properties that refer to some media instances. For concept

Figure 3: A Topia-generated presentation

relationships we define their cardinalities and their inverse relationships.

The AM specifies the navigational structure of the hypermedia presentation in terms of slices and slice relationships. A slice is a meaningful presentation unit that groups concept attributes (from CM) that need to be presented together on the user display. There are two types of slice relationships: compositional relationships (for embedding a slice into another slice) and navigational relationships (as hyperlink abstractions).

Topia

The Topia system generates a broad, general large-scale document structure around media-based RDF resources within an annotated media repository [5]. It then generates a hypermedia presentation based on this structure. Figure 3 shows a Topia-generated presentation.

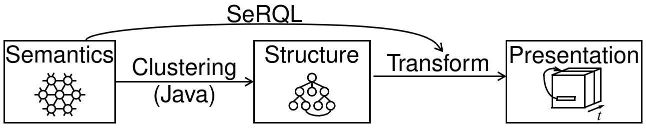

Figure 4: The basics of Topia's architecture

Topia works as shown in Figure 4. First, a user request selects a matching subset from the RDF repository. Then clustering techniques find groups within this subset. This grouping forms the basis for the general document structure, which is used in generating the presentation.

This general document structure is useful for several reasons. One is that it applies to most, if not all, conceptual domains. Another is that it is a structure that has been used in documentation for centuries, making users quite familiar with it. Finally, the interface for navigating this structure is readily recognizable.

Clustering is Topia's primary means of deriving document structure. It finds clusters among the selected RDF resources based on similarities on conceptual proximity [1]. This conceptual proximity can be based on shared properties, placement along measured axes or closeness in a hyperlink node-edge graph.

These domain-independent clustering techniques and presentation devices enable Topia to apply to any domain and provide rudimentary large-scale organization of it into a readily navigable presentation.

Differences

Now that we have given general descriptions of Topia and Hera and their similarities, this section introduces some of their differences. One is the system's domain-independence, with the inverse characteristic of suitability for a specific domain. This relates to the variation in breadth and generality of presentation structure that each system makes, which resulting impacts on layout and interaction.

Domain-dependence. Topia is domain independent while Hera is domain dependent. Hera is domain dependent in that the designer needs to specify the Application Model. Furthermore, the semantic graph in Topia is obtained using AI techniques applied to the RDF resources while the CM (domain schema) is given in Hera. Topia has a general-purpose algorithm for converting semantic graphs to structures progressions. Hera, on the other hand, has designer-specified data transformations by AM on top of CM, which happens at the schema level.

Layout. Earlier work on Topia [5] mentions little about layout, and only simply because some layout needs to occur for there to be a demonstration. Topia's core interest is in generating the structured progression, not in the details of how it is presented. That is, layout is "out of scope", which makes Topia a component of a larger system that does more with layout. Earlier work on Topia discusses issues of how to generate layout and other details of presentation from both underlying RDF with media and a structured progression generated from it. Many of these issues can be encoded in Hera's PM.

Interaction. In Topia, when you have a potentially large broad structure, you need presentation components that are specific to it. Free navigation along the structure at the large-scale level is key. This importance increases as the generation algorithm depends more on automated, domain-independent processing and less on human-designed presentation details and domain specifics. Topia's interactive outline, with is ongoing updating, visibility and navigability, is key to this. The presentation of recurrence in the main display and in the outline is also important here. The ease of navigation along hierarchy, sequence and recurrence are core issues of how the presentation, with its layout, is generated. Other navigation can be added to discourse model-dependent, domain-dependent and directly human-designed components that are more specific and rich in nature. The bridge between Topia and other Hera components enables this.

Broad vs. Fine. In general, we find the differences between Hera and Topia comparable to the role of the cartoon in the creation of the Renaissance-era paintings that dominate our chosen domain: the Rijksmuseum Amsterdam collection. When painting, the artist first makes the cartoon, which is a rough, broad sketch on the blank canvas. Once this cartoon is made, the artist then paints the colors and details over it. Without the cartoon, the painting would lack large-scale composition and consistency. We find Topia's processing analogous to the creation of the cartoon, whereas Hera provides the framework for the rest of the process.

Potential Bridges

Given the distinctions between Hera and Topia the previous section explained, we now explore means of bridging the gap between them. This involves giving means of communication between the systems, allowing each to augment the functions of the other. Topia intentionally leaves presentation details for other components of a broader system to process since Topia is specifically a large-scale presentation structurer. Typically, Hera fills in the localized details under Topia's generality with domains and presentation structure. On the other hand, Hera's generality of process architecture provides more specific input topic selection and output presentation alternatives to Topia's structure-focused processing.

Model Deduction. One potential Hera extension is to use the Topia algorithm to deduce the CM when this is not known, providing schema discovery for use by the designer. This involves generating an absent RDFS from an RDF instance. Another is to have an algorithm that automatically maps CM to AM without knowing the exact concepts in the CM. For example, a concept from CM can have a slice with all its attributes presented and hyperlinks based on its concept relationships, providing domain-independence, such as what Topia already provides.

Clustering the Initial Query Return. The initial query is not important to Topia-based research because Topia deals more with the clustering that happens with the return. Hera's querying mechanism can be the initial query that then applies to Topia clustering, giving the Topia component the larger set of object types, thus making Topia generate richer presentation structure. In terms of the Hera architecture, Hera's integration and data retrieval returns a CM instance. Topia clusters this CM instance, with access to the underlying RDF for clustering and for finding additional concepts to serve as hubs (bases for grouping, resulting in introduction sub-presentations to groups). The Topia component is the broader structuring part of the application design, generating the broader structures in each AM instance.

Increased Domain Specificity. Topia's structured progression is built by an algorithm that maps from the semantical graph to the hierarchy, sequence and recurrence. If the designer is allowed to build structured progressions at the type level based on the schema defined by the identified concept resources in Topia then we will have instance independent structured progression which is similar to the AM in Hera. In this way, Topia becomes domain-dependent, but the quality of the presentation may increase.

Concept Weights. Topia's currently-implemented means of domain-specificity is by providing an interface for assigning significance weights to RDF predicate types, which are the potential types of relations or properties. The algorithm for requesting and processing these is domain-independent, but the designer's input to the interfaces is specific to the domain. This input can also be user-specific, or represent other types of style, which here is applied just to how structure is built. This algorithm from Topia can easily be an XSLT-defined component of the Hera AM.

Impact on Hera's AM. These last two aspects of Topia's contribution, completely domain-independent broad structuring and the significance weights applied to the AM, have interesting interaction with the rest of the AM. There are interesting possibilities for how the established means of specifying AM in Hera can augment Topia's broad structuring and domain-applicability with designer-specific richness at the structural detail level and with detailed components of a specific domain.

XSLT. Topia's clustering is not in XSLT. Some of Topia clustering, particular that involving graph traversal, may not process well in XSLT. This can present a problem in Hera's XSLT-based system. Thus, Hera needs to provide hooks to non-XSLT structuring, such as with the current Topia demo's Java code for clustering. Some XSLT processors provide hooks to external programs, and this could perhaps be used.

Fundamental and Hierarchical Nodes

The previous section discussed some smaller areas of overlap between Hera and Topia, we now begin covering some larger overlaps: that of different types of nodes that both systems handle. Topia and Hera each respond to the user request with a selection from the RDF resources. These items act in both systems as the main content of the presentation. Topia augments this content with related information from the repository that help relate these items to one another. This section formalizes this distinction as part of modeling Topia's processing of it into the broader Hera architecture.

Definitions. Topia considers two types of nodes: fundamental and hierarchical. The fundamental nodes are those returned by the initial query, making up the main content of the presentation. Topia's clustering processing groups these fundamental nodes into a hierarchy. Each internal node in this tree corresponds thus with a hierarchical node.

Screen Displays. Each leaf node in this tree corresponds with a fundamental node. Topia gives a screen display to each of both types of nodes. Hierarchical nodes result in displays that introduce a group before showing its fundamental and lower-level hierarchical nodes. They also generate summary displays after each group.

Hierarchical Nodes in Hera. From the perspective of Topia, Hera uses only fundamental nodes. In Hera, only the nodes are returned from an original query appear in the presentation. Hera joins these nodes into a unified navigation structure in a manner that, compared to Topia, is localized. Thus Topia's contribution to Hera is to find the supplemental hierarchical nodes and build a presentation structure that incorporates both types, thus providing a larger scale unifying navigational structure.

Node Consistency in Topia. The Topia demo currently only searches for matching text in text fields of artifacts, thus returning only artifacts as concept nodes for subsequent clustering. We can easily extend the demo to search text fields of all RDF resources, thus allowing all kinds of conceptual objects to be returned, including genres and places, returning fuller presentations. This would in turn allow artifacts to be hierarchical nodes, thus making available fuller and more varied presentation structure. This amounts to increased exploitation of relationships and the richer semantic graph. The Topia view of the resulting semantic graph is the original underlying RDF with components selected, or highlighted. The selection starts the clustering processes, which refers to the graph as a whole to find additional resources from which the larger structure is formed.

Concept and Presentation Nodes

Another polarity between node types that helps illustrate the Topia and Hera architectures is the divisibility of nodes in both the underlying semantics and the final presentation.

While Topia makes no distinction between these, Hera uses this distinction as part of its adaptation. Related work on the Cuypers system [7] provides additional strategies for mapping between the node sets.

Definitions. Here we consider a concept node to be an RDF resource selected as either a fundamental or hierarchical node for inclusion in the presentation. We define a presentation node as a single unit of composite presentation on the display. That is, a presentation node is a collection of media displayed simultaneously to cooperatively convey one concept or idea. The delimiters between presentation nodes are navigational steps, or "clicks".

Correspondence. Topia has a one-to-one correspondence between concept and presentation nodes. For each concept node, Topia aims to make a single screen display. Hera's slicing, on the other hand, enables multiple presentation nodes to correspond to one concept node. This provides adaptation to devices with varying screen sizes. Smaller devices get multiple, smaller presentation nodes.

Overflow. In earlier work on the Cuypers system, we discuss the strategy of overflow to produce, in the language of this paper, multiple presentation nodes for single concept nodes [6]. If a concept node has too much media for a single display, then Cuypers has it overflow into multiple displays. This overflow can be either as a chain, with a sequence of presentation nodes joined by next and previous buttons, or it can fan out from multiple links in a single, central node to each of multiple nodes.

Granularity. In these manners, both Hera and Cuypers provide means of resolving differences in granularity between concept node media size and presentation node display size. This not only provides adaptation to display size – it also accounts for the potentially wide variation in how much media conveys a concept node, and how big it is. Combining Hera, Cuypers and Topia can provide well-sized presentation nodes that do not necessarily correspond with the structured progression's concept nodes but are reorganized into a meaningful structured progression of the presentation nodes.

mSpace

The mSpace interface to semantic repositories provides rapid user adjustment for a particular type of overriding presentation structure [3]. Its tabular slices of cascading property assignment columns provide readily recognizable overviews and adjustments. This run-time parameter resetting can also apply to the other types of interfaces that Topia and Hera use.

Tabulation. The mSpace system provides a highly interactive tabular interface for browsing semantically defined structure. Topia's consistency component can emulate mSpace's tabular display. Each mSpace column shows different values for one property. Topia's consistency option ensures that siblings groups are each grouped by different values of the same property [1]. While these property types must be the same between siblings in one group, they can vary between groups. Topia can thus emulate mSpace tabulation by having one property type determine the child groups for all groups in the same level of depth of the hierarchy.

mSpace Live Re-parameterization. The contributions of mSpace include not just tabulation but also the increased interaction providing fast, run-time changing of which property type corresponds to each column. Hera and Topia assume more of a "batch mode" in which the user request generates a static presentation in which the only interaction is navigation. mSpace, on the other hand, lets the user change the parameters of how to generate presentation structure while continuing the current presentation, keeping the presentation in its current state but changing the current node's context to the new global presentation structure.

Approaching Live Re-parameterization. Topia and Hera can approach this live parameter adjustment by facilitating the renewed requesting of presentation generation. The Topia demonstrator starts this already by providing buttons from the presentation display to short-cuts for changing parameters. For example, in Topia's global consistency mode, the property type used for global consistency can be changed from each presentation display by a single click on the property name. However, in all cases in the current Hera and Topia demonstrators, such renewed requests start the presentation from the beginning. To approach mSpace's live adjustment, Topia and Hera would need to stay at the current node after such requests. They would also need to maintain information on the user's history within the presentation throughout all live adjustments to the structure generating parameters. Hera is currently extended in this direction in its provision for dynamic adaptation support [4].

Conclusions

In this paper we have discussed the combination of Topia and Hera toward cooperating components of a system for generating hypermedia presentations from media repositories annotated with Semantic Web technologies. We have shown how Hera's architecture allows domain-specific modules that process Topia output or communicate with Topia to state requirements on the output. Comparing the systems brings two polarizations of nodes: fundamental vs. hierarchical and concept vs. presentations. The paper has ended by discussing how Hera and Topia can incorporate mSpace interface features.

From the current applications we see that Hera has breadth of architecture, while Topia has breadth of document structure by having lack of precision or specific discourse. Being neither detailed about the structure nor specific about the discourse allows Topia's breadth of domain application. On the other hand, Hera's tools offer the possibility to have domain specificity, and the combination can therefore benefit from a tool box that offers support for different scenarios.

REFERENCES

1. M. Alberink, L. Rutledge, L. Hardman, and M. Veenstra. Clustering semantics for hypermedia presentation. Technical Report INS-E0403, CWI, 2004.

2. F. Frasincar and G. J. Houben. Hypermedia presentation adaptation on the semantic web. In *Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002*, volume 2347 of *Lecture Notes in Computer Science*, pages 133–142. Springer, 2002.

3. N. Gibbins, S. Harris, and m. schraefel. Applying mspace interfaces to the semantic web. Technical Report EPrint 8639, University of Southampton Electronics and Computer Science, 2003.

4. G. J. Houben, F. Frasincar, P. Barna, and R. Vdovjak. Modeling user input and hypermedia dynamics in hera. In *International Conference on Web Engineering, ICWE 2004*, Lecture Notes in Computer Science. Springer, 2004.

5. L. Rutledge, M. Alberink, R. Brussee, S. Pokraev, W. van Dieten, and M. Veenstra. Finding the story: Broader applicability of semantics and discourse for hypermedia generation. In *ACM Conference on Hypertext and Hypermedia*, pages 67–76. ACM, 2003.

6. L. Rutledge, J. Davis, J. van Ossenbruggen, and L. Hardman. Inter-dimensional Hypermedia Communicative Devices for Rhetorical Structure. In *Proceedings of the International Conference on Multimedia Modeling 2000 (MMM00)*, pages 89–105, 2000.

7. J. van Ossenbruggen, J. Geurts, F. Cornelissen, L. Hardman, and L. Rutledge. Towards second and third generation web-based multimedia. In *The Tenth International World Wide Web Conference*, pages 479–488. ACM, 2001.

8. R. Vdovjak, F. Frasincar, G. J. Houben, and P. Barna. Engineering semantic web information systems in hera. *Journal of Web Engineering*, 2(1-2):3–26, 2003.